Che cosa è un Microcontrollore?

Guida per Studenti

VERSIONE 2.1

Garanzia

La Parallax garantisce i suoi prodotti contro difetti dei materiali e di manifattura per un periodo di 90 giorni dal ricevimento del prodotto. Se scoprite un difetto, Parallax, a suo giudizio, lo riparerà o sostituirà la merce o rimborserà il prezzo di acquisto. Prima di restituire il prodotto a Parallax, richiedete un numero Return Merchandise Authorization (RMA). Scrivere il numero RMA fuori della scatola utilizzata per restituire la merce a Parallax. Per favore includete le seguenti informazioni insieme alla merce restituita: il vostro nome, numero telefonico, indirizzo della spedizione e una descrizione del problema. La Parallax restituirà il vostro prodotto o la sua sostituzione utilizzando lo stesso metodo di spedizione utilizzato per inviare il prodotto a Parallax.

Garanzia soddisfatti o rimborsati di 14 giorni

Se, entro 14 giorni dal ricevimento del vostro prodotto, trovate che non soddisfa le vostre necessità, potete restituirlo per un rimborso completo. La Parallax rimborserà il prezzo di acquisto del prodotto, costi di spedizione/gestione esclusi. Questa garanzia è nulla se il prodotto è stato modificato o danneggiato.

Copyright e marchi

ISBN 1-928982-02-6

Rinuncia di responsabilità

La Parallax, Inc. non è responsabile di danni, speciali, accidentali o conseguenti che derivano da qualsiasi violazione della garanzia o sotto qualsiasi teoria legale, includendo utili persi, tempo di fermo, buonuscita, danno a o sostituzione dell'apparecchiatura o di proprietà o qualsiasi costo di ripristino, riprogrammazione o riproduzione di qualsiasi dato memorizzato in o utilizzato con i prodotti Parallax. La Parallax non è inoltre responsabile di qualsiasi danno personale, tra cui quello alla vita e alla salute, che deriva da utilizzo di un qualunque dei nostri prodotti. Vi assumete la responsabilità completa per la vostra applicazione BASIC Stamp, qualsiasi rischio per la vita possa costituire.

Sito WEB e Liste di Discussione

Il sito web www.parallax.com possiede molti canali per il download, prodotti, applicativi per i clienti, ed acquisti on-line per i componenti usati in questo testo. Manteniamo inoltre diverse liste di discussione e-mail per persone interessate nell'uso dei prodotti Parallax. Queste liste sono accessibili dal sito www.parallax.com attraverso il menù Support ⟶ Discussion Groups. Le liste con cui noi operiamo sono le seguenti:

- BASIC Stamps – Con oltre 2,500 sottoscrittori, questa lista viene largamente utilizzata da tecnici, hobbisti e studenti, che così condividono i loro progetti BASIC Stamp e pongono domande al riguardo.
- Stamps in Class – Creato per educatori e studenti, questa lista ha 500 sottoscrittori che discutono l'uso dei curricula Stamps in Class nei loro corsi. La lista fornisce una opportunità sia per gli studenti che per gli educatori di fare domande ed ottenere risposte.
- Parallax Educators –Esclusivamente per educatori e per coloro che contribuiscono allo sviluppo dello Stamps in Class. Parallax ha creato questo gruppo per avere una risposta ai nostri curricula e fornire un forum e perché gli educatori sviluppino ed ottengano supporto all'insegnamento.
- Parallax Translators – Consistente in meno di 10 persone, lo scopo di questa lista è fornire un canale tra la Parallax e coloro che traducono la nostra documentazione nelle lingue diverse dall'inglese. La Parallax fornisce documenti Word editabili ai nostri traduttori, e cerca di coordinare nel tempo le traduzioni con le pubblicazioni.
- Toddler Robot – Un cliente ha iniziato questa lista di discussione per parlare di applicativi e programmazione del robot Toddler della Parallax.
- SX Tech – Discussione circa la programmazione del microcontrollore SX con strumenti di programmazione assembler Parallax e compilatori BASIC e C di altri fornitori. Circa 600 membri.
- Javelin Stamp – Discussione degli applicativi e progetti con il Javelin Stamp, un modulo Parallax che si programma usando un sottoinsieme del linguaggio di programmazione Java della Sun. Approssimativamente 250 membri.

ERRATA

Sebbene si sia fatto un grande sforzo per assicurare l'accuratezza dei nostri testi, errori possono tuttavia ancora persistere. Se trovate un errore, per favore comunicatecelo inviandoci un e-mail ad editor@parallax.com. Ci sforziamo continuamente di migliorare i nostri materiali educativi e la nostra documentazione, quindi riesaminiamo in continuazione i nostri testi. Saltuariamente, sarà inserita nel nostro sito, www.parallax.com, una pagina di Errata con una lista di errori conosciuti e di correzioni per un determinato testo,. Vi preghiamo di controllare nei download gratuiti, le singole pagine di prodotto per file di errata.

Indice

Prefazione

Questo testo risponde alla domanda "Che cos'è un Microcontrollore?" mostrando agli studenti come possono progettare su misura le proprie intelligenti invenzioni con il BASIC Stamp. Gli Esercizi in questo testo includono un varietà di esperimenti divertenti ed interessanti, pensati per stimolare l'immaginazione dello studente tramite l'uso del movimento, della luce, del suono e della sensazione tattile per introdurre nuovi concetti. Questi Esercizi sono progettati per introdurre lo studente in una varietà di principi base nei campi della programmazione di computer, elettricità ed elettronica, matematica e fisica. Molti degli Esercizi presentano un facile apprendimento di pratiche di progettazione usate dagli ingegneri e tecnici nella creazione di moderne macchine ed applicazioni, con l'uso di parti economiche e di facile reperibilità.

UDITORIO

Questo testo è organizzato cosicché possa essere usato dalla più ampia varietà possibile di studenti, come anche da autodidatti. Gli studenti della scuola media, possono provare gli esempi in questo testo in maniera guidata semplicemente seguendo le istruzioni passo-passo, con la supervisione di un istruttore. All'altro estremo, le abilità di soluzione dei problemi per studenti pre-laurea, saranno messe alla prova dagli Esercizi finali di ciascun capitolo e dai compiti assegnati nella sezione progetti alla fine di ogni capitolo. Gli autodidatti potranno imparare al loro giusto ritmo ed ottenere assistenza attraverso il forum Stamps in Class citato sotto.

SUPPORTO E GUIDA PER GLI INSEGNANTI

Sono disponibili per coloro che vorranno essere supportati nell'uso di questo testo i seguenti gruppi di discussione Parallax Yahoo!:

Gruppo Stamps In Class: Aperto a studenti, educatori ed autodidatti, questo forum permette ai membri di farsi domande reciprocamente e condividere le risposte man mano che affrontano Esercizi e progetti in questo testo.

Gruppo Educatori Parallax: La guida per gli insegnanti è disponibile solo per gli educatori; può essere ottenuta attraverso questo forum dopo che il proprio stato di educatori sia stato verificato dalla Parallax. Questo forum limitato fornisce supporto per educatori, e poiché lo sviluppo del curriculum Stamps in Class è continuo incoraggia il ritorno di informazioni.

Questi gruppi sono accessibili tramite il gruppo di discussione nel menu supporto del sito www.parallax.com. Se avete difficoltà nella sottoscrizione di uno qualsiasi di questi gruppi Yahoo!, oppure avete altre domande circa questo testo, oppure circa lo Stamps in Class, contattate il team Parallax Stamps in Class direttamente tramite la seguente E-Mail stampsinclass@parallax.com.

IL CURRICULUM STAMPS IN CLASS

Che cosa è un Microcontrollore? È la porta d'accesso al nostro curriculum Stamps in Class. Dopo aver completato questo testo, potrete continuare i vostri studi con qualsiasi delle guide per studenti sotto elencate. Tutti i libri in elenco, sono disponibili per un download gratuito al sito www.parallax.com. Le versioni sotto citate, sono le edizioni aggiornate alla data di questa ristampa. Vogliate controllare i nostri siti www.parallax.com oppure www.stampsinclass.com per le ultime revisioni; ci sforziamo continuamente per migliorare il nostro programma educativo.

Guide per gli Studenti Stamps in Class:

Per una buona comprensione delle pratiche di progettazione dei dispositivi e dei macchinari moderni, è altamente raccomandato compiere gli Esercizi ed i progetti delle seguenti guide per studenti.

"*Applied Sensors*", Student Guide, Version 2.0, Parallax Inc., 2003
"*Basic Analog and Digital*", Student Guide, Version 2.0, Parallax Inc., 2003
"*Industrial Control*", Student Guide, Version 2.0, Parallax Inc., 2002
"*Robotica!*", Guida per lo Studente, Versione 1.5, Parallax Inc., 2000

Altri Kit di Robotica:

Alcuni arrivano ai curriculum Stamps in Class tramite la guida per studenti *Robotica!*. Dopo averla completata, sarete pronti per affrontare i seguenti testi e kit di robotica più avanzati:

"*Advanced Robotics: with the Toddler*", Student Guide, Version 1.2, Parallax Inc., 2003
"*SumoBot*", Student Guide, Version 1.1, Parallax Inc., 2002

Kit per Progetti Educativi:

Elements of Digital Logic e *Understanding Signals* focalizzano più attentamente su argomenti elettronici, mentre *StampWorks* fornisce una quantità di progetti utili per gli hobbisti, gli inventori e i progettisti interessati nello sperimentare svariati progetti.

"*Elements of Digital Logic*", Student Guide, Version 1.0, Parallax Inc., 2003
"*StampWorks*", Manual, Version 1.2, Parallax Inc., 2001
"*Understanding Signals*", Student Guide, Version 1.0, Parallax Inc., 2003

Riferimenti

Questo libro è un riferimento essenziale per tutte le guide per studenti Stamps in Class. Esso è pieno di informazioni, sui microcontrollori BASIC Stamp, la Board of Education le altre nostre schede madri, il nostro Editor BASIC Stamp, ed il nostro linguaggio di programmazione.

"*Manuale del BASIC Stamp*", Users Manual, Version 2.0c, Parallax Inc., 2000

CONDIZIONI PER LA DUPLICAZIONE

Questo testo è disponibile in forma stampata presso la Parallax, Inc. A motivo della stampa in grandi quantità, spesso il prezzo all'utente finale è inferiore ai costi di duplicazione commerciale.

TRADUZIONE IN ALTRE LINGUE

I testi educativi della Parallax possono essere tradotti in altre lingue con il nostro consenso (e-mail stampsinclass@parallax.com). Se decidete di fare qualsiasi traduzione vogliate prendere contatto con noi cosi ché vi possiamo fornire la documentazione correttamente formattata in MS Word. Manteniamo inoltre un gruppo di discussione per i traduttori Parallax a cui potete associarvi. Si chiama Parallax Translators Yahoo-group, e

nelle prime pagine di questo testo, sono fornite informazioni per accedervi. Vedi la sezione intitolata:

Sito WEB e Liste di **Discussione**
Il sito web www.parallax.com possiede molti canali per il download, prodotti, applicativi per i clienti, ed acquisti on-line per i componenti usati in questo testo. Manteniamo inoltre diverse liste di discussione e-mail per persone interessate nell'uso dei prodotti Parallax. Queste liste sono accessibili dal sito www.parallax.com attraverso il menù Support ⟶ Discussion Groups. Le liste con cui noi operiamo sono le seguenti:

- BASIC Stamps – Con oltre 2,500 sottoscrittori, questa lista viene largamente utilizzata da tecnici, hobbisti e studenti, che così condividono i loro progetti BASIC Stamp e pongono domande al riguardo.
- Stamps in Class – Creato per educatori e studenti, questa lista ha 500 sottoscrittori che discutono l'uso dei curricula Stamps in Class nei loro corsi. La lista fornisce una opportunità sia per gli studenti che per gli educatori di fare domande ed ottenere risposte.
- Parallax Educators –Esclusivamente per educatori e per coloro che contribuiscono allo sviluppo dello Stamps in Class. Parallax ha creato questo gruppo per avere una risposta ai nostri curricula e fornire un forum e perché gli educatori sviluppino ed ottengano supporto all'insegnamento.
- Parallax Translators – Consistente in meno di 10 persone, lo scopo di questa lista è fornire un canale tra la Parallax e coloro che traducono la nostra documentazione nelle lingue diverse dall'inglese. La Parallax fornisce documenti Word editabili ai nostri traduttori, e cerca di coordinare nel tempo le traduzioni con le pubblicazioni.
- Toddler Robot – Un cliente ha iniziato questa lista di discussione per parlare di applicativi e programmazione del robot Toddler della Parallax.
- SX Tech – Discussione circa la programmazione del microcontrollore SX con strumenti di programmazione assembler Parallax e compilatori BASIC e C di altri fornitori. Circa 600 membri.

Javelin Stamp – Discussione degli applicativi e progetti con il Javelin Stamp, un modulo Parallax che si programma usando un sottoinsieme del linguaggio di programmazione Java della Sun. Approssimativamente 250 membri dopo la copertina.

CONTRIBUTORI SPECIALI

Il Gruppo di lavoro che la Parallax ha messo insieme per scrivere questo testo comprende: progettazione del curriculum e scritti tecnici a cura di Andy Lindsay, illustrazioni di Rich Allred, Copertina di Jen Jacobs e Larissa Crittenden, consulenza generale a cura Aristides Alvarez e Jeff Martin, consulenza elettromeccanica a cura di John Barrowman, revisione tecnica di Kris Magri, edizione tecnica di Stephanie Lindsay, commissione dei revisori Rich Allred, Gabe Duran, Stephanie Lindsay, e Kris Magri.

La guida per studenti *Che cos'è un Microcontrollore?* Versione 2.1è stata scritta da Andy Lindsay dopo aver raccolto le osservazioni e le risposte degli insegnanti ai corsi Parallax per insegnanti in tutta la nazione. Andy ha studiato ingegneria elettrica ed elettronica all'università statale della California, Sacramento, e questa è la terza edizione della guida per studenti Stamps in Class. Fornisce inoltre il suo contributo a diverse dispense sui

Microcontrollori per corsi pre-laurea. Quando non è impegnato a scrivere materiale educativo, Andy lavora all'ingegnerizzazione di prodotto per la Parallax.

La Parallax desidera ringraziare, il membro del gruppo StampsInClass Yahoo Robert Ang per la completa revisione delle bozze e per le indicazioni dettagliate, il tecnico esperto e stimato cliente Sid Weaver per la sua revisione approfondita. Grazie anche agli autori Stamps in Class, Tracy Allen (*Applied Sensors*), e Martin Hebel (*Industrial Control*) per la loro revisione e le loro raccomandazioni. Andy Lindsay vuole ringraziare suo padre Marshall e suo cognato Kubilay per la loro consulenza musicale e per i loro suggerimenti. Stamps in Class è stata fondata da Ken Gracey, e Ken vuole ringraziare lo staff Parallax per il grande lavoro che fanno. **Un ringraziamento anche a Stefano Caruso per la traduzione di questo testo in Italiano**. Tutti e ciascun Parallaxiano hanno contribuito a questo come a tutti i testi Stamps in Class.

Capitolo #1: INIZIARE

QUANTI MICROCONTROLLORI AVETE USATO OGGI?

Un microcontrollore è il genere di computer in miniatura che potete trovare in ogni genere di oggetti. Alcuni esempi di oggetti comuni di tutti i giorni che contengono microcontrollori sono illustrati in Figura 1-1. Se possiede dei tasti ed un display, è probabile che abbia anche un cervello a microcontrollore.

Figura 1-1
Esempi di oggetti comuni che contengono un Microcontrollore.

Provate a fare un elenco e contate quanti dispositivi con microcontrollore usate in una giornata tipo. Qui ci sono alcuni esempi: se al mattino la vostra radio si spenge, e premete più volte il tasto " snooze", la prima cosa che fate nella vostra giornata è interagire con un microcontrollore. Riscaldare dei cibi nel forno a microonde oppure fare una telefonata con il cellulare, implica usare dei microcontrollori. E questo è solamente l'inizio. Qui ci sono alcuni altri esempi: accendere un televisore con il telecomando, giocare con un videogioco, usare una calcolatrice e guardare l'ora sul vostro orologio da polso digitale. Tutti questi dispositivi contengono dei microcontrollori che interagiscono con voi.

IL BASIC STAMP 2 – IL VOSTRO NUOVO MICROCONTROLLORE

Il modulo BASIC Stamp 2 mostrato in Figura 1-2 ha un microcontrollore al suo interno. E' il circuito integrato nero la scritta "PIC16C57". Il resto dei componenti nel BASIC Stamp si trovano anche negli oggetti comuni che usate tutti i giorni. Tutto l'insieme è chiamato sistema computerizzato entrocontenuto. Questo nome è comunemente accorciato in "computerizzato". Gli Esercizi in questo testo, vi guideranno nella costruzione di circuiti simili a quelli presenti negli oggetti di uso comune e negli apparati ad alta tecnologia. Scriverete inoltre programmi al computer che il BASIC Stamp eseguirà. Questi programmi osserveranno e faranno controllare dal BASIC Stamp I circuiti cosicché essi svolgano funzioni utili.

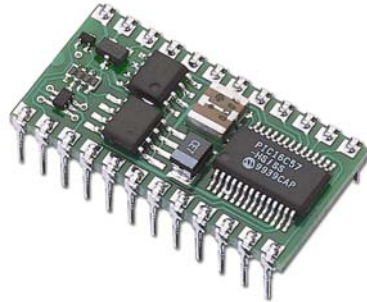



Figura 1-2
Il BASIC Stamp 2

In questo testo, BASIC Stamp è usato per indicare il BASIC Stamp 2. Ci sono altri BASIC Stamp, alcuni dei quali sono mostrati nella Figura 1-3. Ciascun BASIC Stamp è codificato con un colore. Il BASIC Stamp 2 è verde. Il BASIC Stamp 2e è rosso. Il BASIC Stamp 2 SX è blù, ed il BASIC Stamp 2p è oro. Ciascuna variante del BASIC Stamp 2 è leggermente differente, maggiore velocità, più memoria, funzionalità aggiunte, oppure una combinazione di queste funzioni aggiuntive.

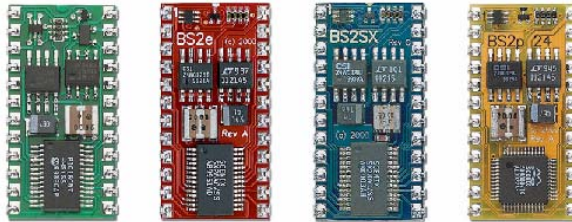



Figura 1-3
Moduli BASIC Stamp

Da sinistra a destra: BASIC Stamp 2, 2e, 2SX, e 2p.

SORPRENDENTI INVENZIONI CON IL BASIC STAMP

Le applicazioni comuni non sono le sole cose che contengono microcontrollori. Robot, macchinari, progetti aerospaziali ed altri dispositivi ad alta tecnologia sono fabbricati anche con microcontrollori. Diamo uno sguardo ad alcuni esempi che sono stati creati con i BASIC Stamp.

I Robot sono stati progettati per fare qualsiasi cosa, dall'aiutare gli studenti ad apprendere di più circa i microcontrollori, a rasare i prati, alla soluzione di problemi meccanici complessi. La Figura 1-4 mostra due robot esempio. Su ciascuno di questi robot, gli studenti usano il BASIC Stamp per leggere sensori, controllare motori, e comunicare con altri computer. Il robot sulla sinistra è chiamato Boe-Bot. Il progetto che è pubblicato nel libro *Robotica!* Può essere approfondito usando il Boe-Bot dopo che avete eseguito gli

Esercizi di questo testo. Il robot sulla destra è stato costruito da un gruppo di studenti ed iscritto in un'importante gara di robotica. Il fine della competizione è diverso ogni anno. Nell'esempio mostrato, lo scopo era scegliere quale gruppo di robot era più veloce nell'ordinare tempo le ciambelle colorate.

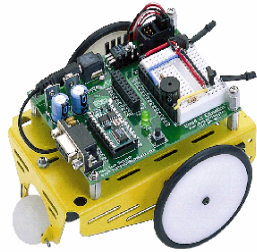



Figura 1-4
Robot Educazionali

Il Boe-Bot Parallax (a sinistra)
First Competition Robot (a destra)

Altri robot risolvono problemi complessi, come il robot per il volo autonomo illustrato a sinistra nella Figura 1-5. Questo robot è stato costruito e collaudato dagli studenti di ingegneria meccanica dell'università di Irvine, California. Hanno usato un BASIC Stamp per la comunicazione con un satellite del sistema GPS cosicché il robot possa conoscere la sua posizione ed altitudine. Il BASIC Stamp inoltre legge i sensori di livellamento e controlla i motori perché il robot possa volare correttamente. Il robot millepiedi meccanico, sulla destra, è stato sviluppato da un professore all'università tecnica Nanyang di Singapore. Ha più di 50 BASIC Stamp, e ciascuno comunica con gli altri tramite una rete complessa che aiuta a controllare ed orchestrare il movimento di ciascun set di zampe. Non solamente robot come questi ci aiutano a capire meglio i progetti della natura, ma potrebbero eventualmente essere usati per esplorare località remote o perfino altri pianeti.

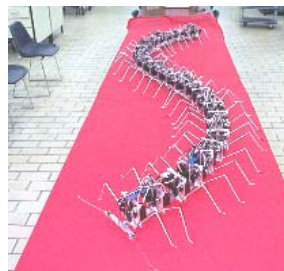



Figura 1-5
Esempi di robot sperimentali che contengono Microcontrollori

Robot volante autonomo, Università Irvine CA (a sinistra) e progetto Millepiede all'università Nanyang (a destra)

Con l'aiuto dei microcontrollori, i robot si assumeranno anche i compiti giornalieri, come per esempio tagliare l'erba. Il BASIC Stamp all'interno del tagliaerba robotizzato della figura 1-6 fa sì che rimanga all'interno dei confini del prato, legge anche I sensori che rilevano ostacoli e controlla I motori che lo fanno muovere.

Figura 1-6
Tagliaerba robotizzato

Prototipo del Robot Shop

I microcontrollori sono usati anche nei progetti scientifici, ad alta tecnologia, ed aerospaziali. La stazione meteorologica illustrata a sinistra nella Figura 1-7 viene usata per raccogliere dati ambientali della degradazione della barriera corallina. Il BASIC Stamp al suo interno raccoglie dati da una varietà di sensori e li memorizza per un recupero successivo da arte degli scienziati. Il sottomarino al centro, è un veicolo per l'esplorazione sottomarina, e le sue eliche, telecamere e luci, sono tutte controllate da BASIC Stamp. Il razzo mostrato sulla destra fa parte di una gara per lanciare nello spazio un razzo privato. Nessuno ha vinto la gara, ma questo razzo ha almeno partecipato perché è stato lanciato da una piattaforma portata a grande altezza da palloni meteorologici. Il BASIC Stamp controllava esattamente ogni aspetto del lancio, dalla sequenza di lancio rilevando l'altitudine, comunicava informazioni ai tecnici della base a terra, ed ha attuato effettivamente lancio.

Figura 1-7
Esempi di Microcontrollori per alta tecnologia aerospaziale
Raccoglitore di dati ecologici della EME Systems (a sinistra), Ricerche sottomarine della Harbor Branch Institute (al centro), e lancio di prova della JP Aerospace (destra)

Dalle applicazioni domestiche comuni su su fino alle applicazioni scientifiche ed aerospaziali, i basilari dei microcontrollori che vi servono per cominciare progetti come questi, saranno introdotti qui. Lavorando sugli Esercizi di questo libro, sperimenterete ed apprenderete come usare una moltitudine di blocchi basilari presenti in tutti questi oggetti. Costruirete circuiti per display, sensori, e controlli di motori. Apprenderete come collegare questi circuiti al BASIC Stamp, ed a scrivere programmi che gli faranno controllare i display, raccogliere dati dai sensori, e controllare il movimento. Conseguentemente, imparerete molti concetti e tecniche di elettronica e programmazione. Prima di aver finito il corso, potreste trovarvi ad aver inventato un oggetto di vostra progettazione.

HARDWARE E SOFTWARE

Ambientarsi con il BASIC Stamp è simile all'ambientamento con un PC od un portatile nuovi. La prima cosa che molte persone devono fare quando comprano un PC od un portatile nuovo, è di toglierlo dalla scatola, collegarlo, installare e provare i programmi e può darsi, perfino scrivere del software di propria mano usando un linguaggio di programmazione. Se questa è la vostra prima volta nell'uso del BASIC Stamp, dovrete fare gli stessi Esercizi. Se siete in una classe, potete trovare il BASIC Stamp già assemblato. Se questo è il caso, il vostro insegnante può fornirvi altre istruzioni. Se no, gli Esercizi in questo capitolo vi guideranno in tutte le fasi di assemblaggio ed accensione del vostro nuovo BASIC Stamp.

ESERCIZIO #1: PROCURARSI IL SOFTWARE

Il software che userete nella maggioranza degli Esercizi e dei progetti di questo testo, è l'Editor BASIC Stamp (versione 2.0 o superiore). Userete questo software per scrivere programmi che il BASIC Stamp eseguirà. Potrete anche usare questo software per visualizzare messaggi inviati dal BASIC Stamp che vi aiuteranno a capire che cosa sta accadendo.

L'Editor BASIC Stamp è un programma gratuito, e le due maniere più facili per averlo sono:

- **Scaricarlo da Internet:** Cercare per "BASIC Stamp Windows Editor version 2.0…" sulla pagina www.parallax.com → Downloads → BASIC Stamp Software.
- **Incluso nel CD Parallax:** Seguire le indicazioni nella pagina di benvenuto. Assicuratevi che la data stampata sul CD sia Maggio 2003 o successivo.

Avete fretta? Prendete la vostra copia di Editor del BASIC Stamp versione 2.0 (o successiva) ed installatela sul vs PC o laptop. Andate quindi all'**Esercizio** #1: assemblaggio dell'hardware e collaudo del sistema.

Se avete altre domande, l'Esercizio **#1** può essere usato come una guida passo-passo per procurarsi il software, e l'**Esercizio #2** può essere usato come una guida per la procedura di installazione.

Requisiti di Sistema

Per far girare l'Editor BASIC Stamp vi servirà un PC o anche un laptop. L'apprendimento con il BASIC Stamp sarà più facile se il vostro computer PC o laptop, ha le seguenti caratteristiche:

- Sistema operativo Windows 95 o successivi
- Una porta seriale o USB
- Un drive CD-ROM, l'accesso al World Wide Web, o ambedue

Adattatore per porta USB se il vostro computer ha solamente porte USB, vi servirà un adattatore da USB a Seriale. Per i dettagli e le istruzioni sull'installazione, vedi l'Appendice A: Adattatore USB <> Seriale.

Scaricare il Software da Internet.

Scaricare l'Editor BASIC Stamp dal sito web della Parallax. La pagina web mostrata in Figura 1-8 può essere diversa quando visitate il sito. Ciononostante la procedura per scaricare il software dovrebbe essere similare alla:

√ Usando un browser web, andare a www.parallax.com (mostrato in Figura 1-8).
√ Puntare al menù *Downloads* per mostrare le opzioni.
√ Puntare al *software BASIC Stamp* e selezionarlo.

Figura 1-8
Il sito Web:

www.parallax.com

√ Quando siete nella pagina Software BASIC Stamp, trovate la versione più recente dell'Editor Windows BASIC Stamp, una versione 2.0 o successiva.

√ Clickate l'icona *Download*. Nella Figura 1-9, l'icona download è simile ad una cartella a destra della descrizione: "BASIC Stamp Windows Editor version 2.0 Beta 1 (6MB)".

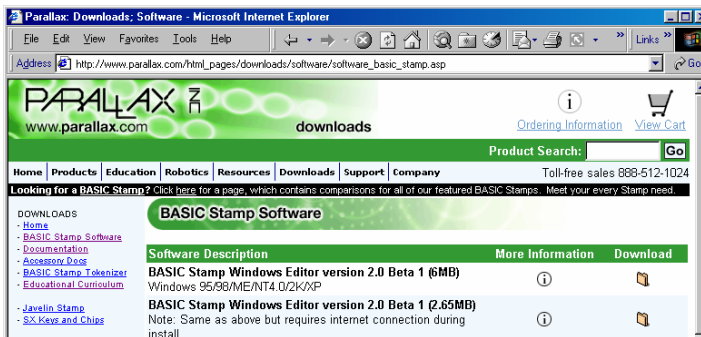



Figura 1-9
La pagina Downloads del sito Web Parallax

√ Quando appare la finestra mostrata in Figura 1-10, selezionate: *Save this program to disk.*

√ Clickare il tasto *Ok*.

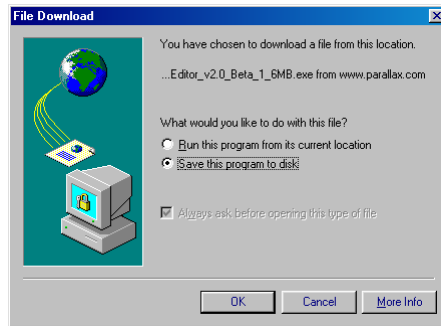



Figura 1-10
Finestra di download dei File.

La Figura 1-11 mostra la finestra "Salva come". Potete usare il campo *Save per* cercare sul vostro disco rigido una cartella in cui memorizzare il file.

√ Dopo aver scelto dove salvare il file, clickare il tasto *Save*.

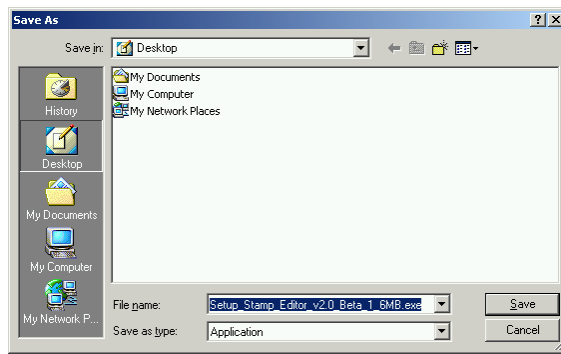



Figura 1-11
Finestra "Salva come"

Trovare una posizione dove memorizzare il file

√ Attendere che il programma di installazione dell'Editor BASIC Stamp (mostrato nella Figura 1-12) sia stato scaricato. Se state usando un collegamento via modem, ciò può richiedere diverso tempo.

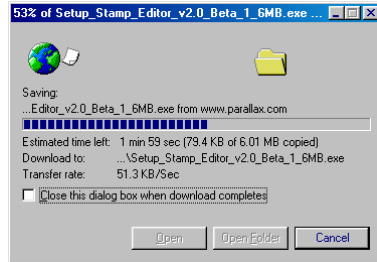



Figura 1-12
Finestra di Avanzamento del Download

√ Quando il Download è completo, lasciare aperta la finestra mostrata in Figura 1-13 mentre andate alla sezione seguente - Esercizio #2: Installazione del Software.

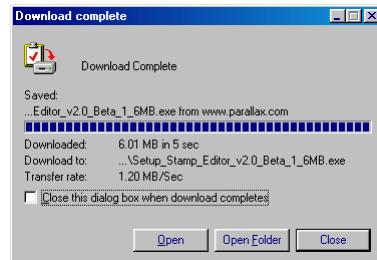



Figura 1-13
Download Completato.

Andare all'Esercizio #2: Installazione del Software.

Altri download gratuiti nel sito web della Parallax includono:

- Questo testo e altri testi Stamps in Class.
- Video di Robot
- Ulteriore software gratuito
- Centinaia di applicazioni ed esperimenti che potete provare.

Trovare il Software sul CD Parallax.

Potete installare l'Editor BASIC Stamp anche dal CD Parallax, ma il CD deve essere marcato Maggio 2003 o posteriore perché la versione dell'editor sia compatibile con gli esempi di questo testo. Potete trovare l'anno ed il mese del CD Parallax esaminando l'etichetta sul CD.

√ Posizionare il CD Parallax nel drive del vostro computer. Il browser del CD Parallax viene chiamato applicazione di Benvenuto. Viene mostrata nella Figura 1-14 e dovrebbe attivarsi in seguito all'introduzione del CD nel drive del vostro computer.

√ Se l'applicazione non parte automaticamente, fate doppio-click su *My Compute*r, quindi doppio-click sull'icona del drive CD, quindi doppio-click su *Welcome*.

√ Clickare il collegamento *Software* mostrato in Figura 1-14.

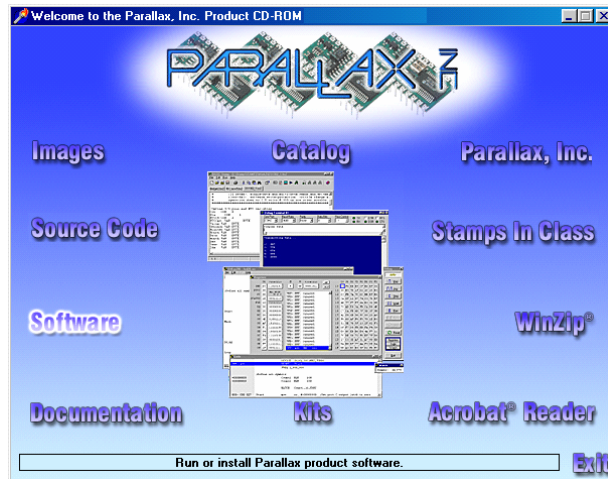



Figura 1-14
Il Browser del CD Parallax.

√ Clickare il + vicino alla cartella BASIC Stamps mostrata in Figura 1-15.

√ Clickare il + vicino alla cartella Windows.

√ Clickare l'icona del dischetto floppy

√ Con l'etichetta "Stamp 2/2e/2sx/2p/2pe (stampw.exe)".

√ Andare all'Esercizio #2: Installazione del Software.

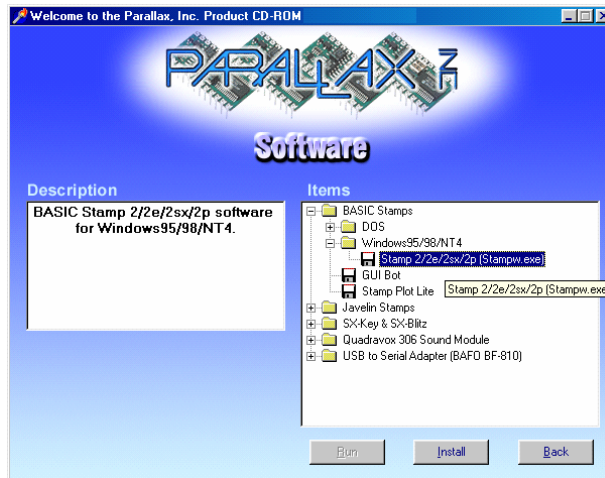



Figura 1-15
Il Browser del CD Parallax.

Selezionare dalla pagina Software, il programma di installazione dell'Editor del BASIC Stamp.

I download gratuiti, disponibili nel sito web Parallax sono inclusi nel CD Parallax, ma soltanto fino alla data di creazione del CD. La data sul fronte del CD indica quando è stato creato. Se il CD è vecchio solo di pochi mesi, è probabile che abbiate la versione più aggiornata. Se è una versione vecchia, prendete in considerazione la possibilità di richiedere una versione più aggiornata alla Parallax o di scaricare I file di cui avete bisogno dal sito web Parallax.

ESERCIZIO #2: INSTALLAZIONE DEL SOFTWARE

A questo punto, avete, scaricato l'installatore dell'Editor BASIC Stamp dal sito web Parallax, oppure lo avete localizzato sul CD della Parallax. Ora è tempo di installarlo.

Installazione del Software passo-passo

√ Se avete scaricato l'Editor BASIC Stamp da Internet, clickare sul bottone *Open Folder* nella finestra "Download Complete" mostrata in Figura 1-16.

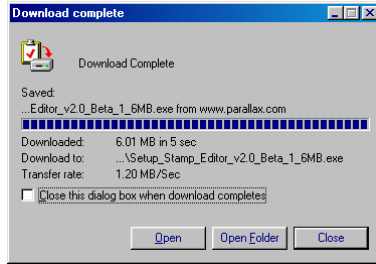



Figura 1-16
Finestra Download Complete.

Se siete arrivati a questo punto dalla sezione "Scaricare il sofware da Internet", clickare sul bottone Open Folder.

√ Se avete localizzato il software sul CD Parallax, clickare il bottone *Install* mostrato in Figura 1-17.

Figura 1-17
Il Browser del Parallax CD

Il bottone Install è posizionato in basso nella finestra.

√ Quando la finestra InstallShield Wizard dell'Editor del BASIC Stamp si è aperta, clickare sul bottone *Next* mostrato in Figura 1-18.

Figura 1-18
InstallShield Wizard per l' Editor del BASIC Stamp

Clickare Next.

√ Per il setup type selezionare *Typical* come mostrato in Figura 1-19.

1

√ Clickare il bottone *Next*.

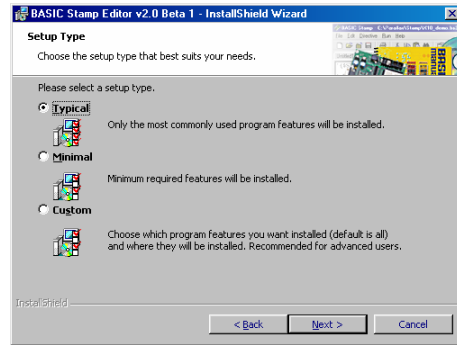



Figura 1-19
Setup Type

Clickare Typical, quindi clickare il bottone Next.

√ Quando InstallShield Wizard vi dice “Ready to Install the Program”, clickare il bottone *Install* mostrato in Figura 1-20.

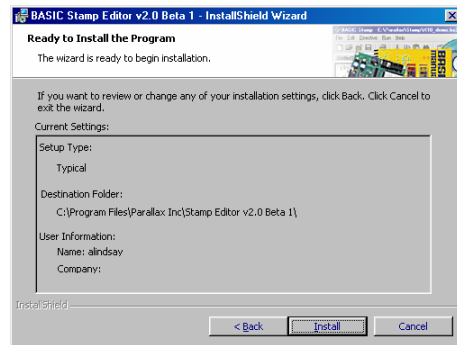



Figura 1-20
Ready to Install.

Clickare il bottone Install.

√ Quando la finestra InstallShield Wizard vi dice “InstallShield Wizard Completed”, come mostrato in Figura 1-21, clickare *Finish*.

Figura 1-21
InstallShield Wizard Completed:

Clickare il bottone Finish.

ESERCIZIO #3: ASSEMBLAGGIO DELL'HARDWARE E COLLAUDO DEL SISTEMA

Il BASIC Stamp per funzionare, richiede di essere collegato ad un alimentatore. Richiede anche di essere collegato ad un PC in modo che possa essere programmato. Dopo aver fatto queste connessioni, per collaudare il sistema potete usare l'Editor del BASIC Stamp. Questo Esercizio, vi mostrerà come farlo.

Il BASIC Stamp, la Board of Education e la HomeWork Board

Il BASIC Stamp 2 e la Board of Education sono illustrate in Figura 1-22. Come citato in precedenza, il BASIC Stamp è un computer molto piccolo. Questo computer molto piccolo s'inserisce nella Board of Education, che è chiamata scheda madre. Come presto vedrete, la Board of Education rende facile la connessione al BASIC Stamp di un alimentatore ed un cavo seriale. Negli Esercizi successivi, vedrete inoltre come la Board of Education faciliti la costruzione di circuiti e la loro connessione al BASIC Stamp.

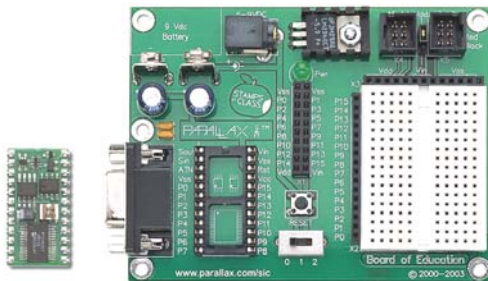



Figura 1-22
BASIC Stamp 2 (a sinistra) e Board of Education (a destra)

La scheda BASIC Stamp HomeWork è illustrata in Figura 1-23. Questa scheda è simile alla Board of Education, ma con il BASIC Stamp 2 costruito al suo interno. Per gli Esercizi di questo testo, potete usare sia la Board of Education che la scheda BASIC Stamp HomeWork.

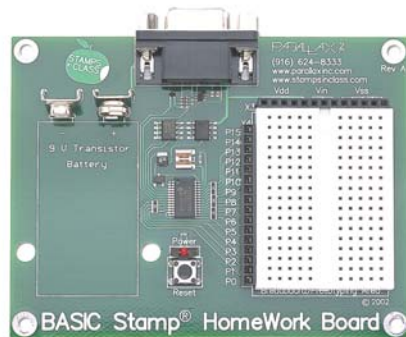



Figura 1-23
Scheda BASIC Stamp HomeWork

Per Maggiori Informazioni circa le prestazioni, i componenti e le funzioni del BASIC Stamp, Board of Education, e HomeWork Board. Vedere l'Appendice C: BASIC Stamp e Scheda Madre Componenti e Funzioni a pagina 309.

Hardware Richiesto

(1) BASIC Stamp 2 e (1) Board of Education
- oppure -
(1) Scheda BASIC Stamp HomeWork

Vedere la Figura 1-24:
(1) pila da 9-V
(1) Striscia con quattro piedini adesivi in gomma
(1) Cavo Seriale

Figura 1-24
Pila da 9 V,
Piedini in gomma, e
Cavo Seriale

Cominciare con una batteria da 9 V nuova o completamente carica. Evitare la confusione che una batteria esaurita può causare. Iniziare con una pila alcalina o con una batteria ricaricabile che sia stata appena ricaricata.

ATTENZIONE

Prima di usare un adattatore da rete "sostitutore di pile", o un alimentatore in corrente continua:

√ Consultare l'**Error! Reference source not found.** a pagina **Error! Bookmark not defined.** per assicurarvi che l'alimentazione che usate, sia adatta agli Esercizi di questo testo.

Collegare l'Hardware

Sia la Board of Education che la scheda BASIC Stamp HomeWork sono fornite di una striscia con quattro piedini adesivi in gomma. Questi piedini in gomma sono illustrati nella Figura 1-25, e dovrebbero essere attaccati sul lato inferiore della Board of Education o della scheda BASIC Stamp HomeWork.

Figura 1-25
Piedini in gomma

√ La Board of Education che state usando, ha dei cerchietti disegnati sul lato inferiore per mostrare dove ciascun piedino in gomma deve essere messo. Rimuovere ciascun piedino in gomma dalla striscia adesiva e attaccatelo sul lato inferiore della Board of Education come mostrato in Figura 1-26.

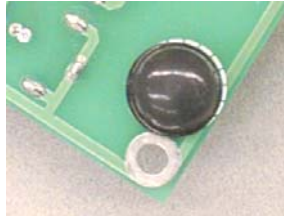



Figura 1-26
Piedini in gomma attaccati sul lato inferiore della Board of Education

√ Se state usando la scheda BASIC Stamp HomeWork, rimuovere ciascun piedino in gomma dalla striscia adesiva e attaccatelo sul lato inferiore della scheda HomeWork, vicino a ciascun foro metallizzato a ciascun angolo della scheda come mostrato nella Figura 1-27.

Figura 1-27
Piedini in gomma attaccati sul lato inferiore della scheda HomeWork

Ora, la Board of Education o la scheda BASIC Stamp HomeWork deve essere collegata con un cavo seriale al vostro PC o laptop.

√ Collegare il cavo seriale ad una porta seriale libera sul retro del vostro computer come mostrato nella Figura 1-28.

Adattatore di porta USB Se state usando un adattatore da seriale ad USB:

√ Collegare il lato USB dell'adattatore alla porta USB del vostro PC.

√ Collegare il lato seriale dell'adattatore o direttamente alla vostra Board of Education o scheda HomeWork, oppure collegatelo al cavo seriale come mostrato in Figura 1-28.

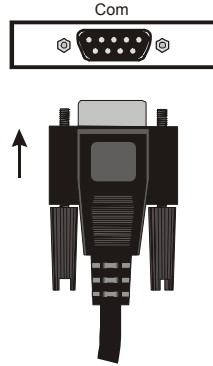



Figura 1-28
Porta seriale di PC o Laptop

Inserire il connettore del cavo seriale in una porta disponibile del vostro PC o laptop.

Se state usando il BASIC Stamp 2 e la Board of Education:

√ Posizionare l'interruttore a 3 posizioni della Board of Education in posizione 0 come mostrato in Figura 1-29.

Figura 1-29
Interruttore a 3 posizioni

Posizionare a 0 per togliere corrente.

Solamente la Board of Education Rev C ha un interruttore a 3 posizioni. Per spengere la Board of Education Rev B, Semplicemente scollegate lo spinotto dell'alimentatore o togliete la pila come mostrato in Figura 1-30, passo 3 o 4.

√ Se il vostro BASIC Stamp non è già inserito nella Board of Education, inseritelo nello zoccolo mostrato Figura 1-30, passo 1. Assicuratevi che i piedini siano allineati correttamente con i fori dello zoccolo, quindi premere fino in fondo.
√ Collegare il cavo seriale nella Board of Education come indicato al passo 2.
√ Collegare un alimentatore nel connettore marcato 6-9 VDC come indicato al passo 3, o inserite una pila da 9-V nel connettore come indicato al passo 4.
√ Muovere l'interruttore a 3 posizioni dalla posizione 0 alla posizione 1.
√ La luce verde etichettata Pwr sulla Board of Education si deve accendere.

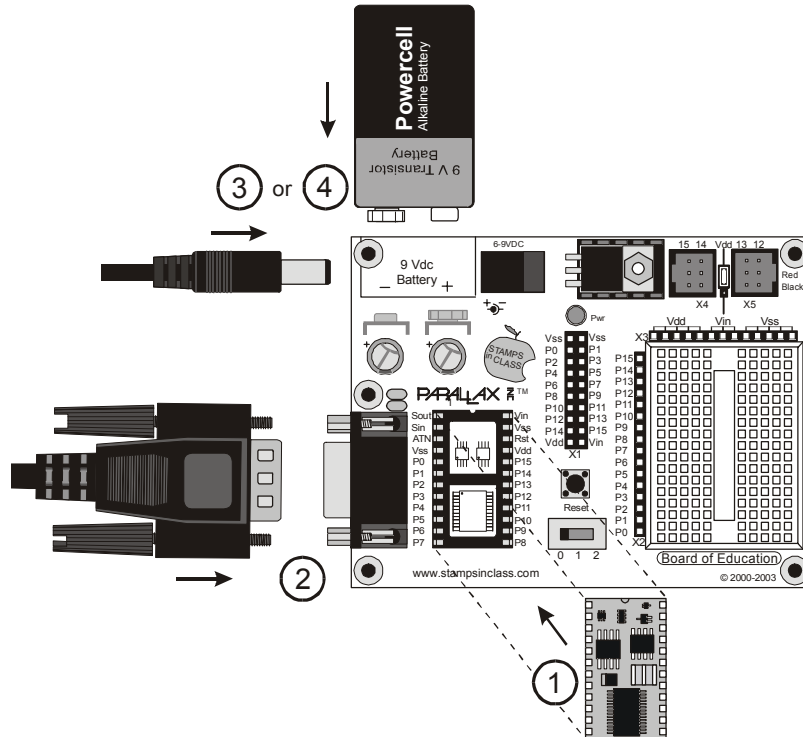



Figura 1-30
Board of Education, BASIC Stamp, Pila e cavo seriale.

Collegare I componenti nell'ordine indicato nello schema.

Se state usando la scheda BASIC Stamp HomeWork:

√ Collegare il cavo seriale alla scheda HomeWork come indicato in Figura 1-31, al passo 1.

√ inserite una pila da 9-V nel connettore come indicato al passo 2.

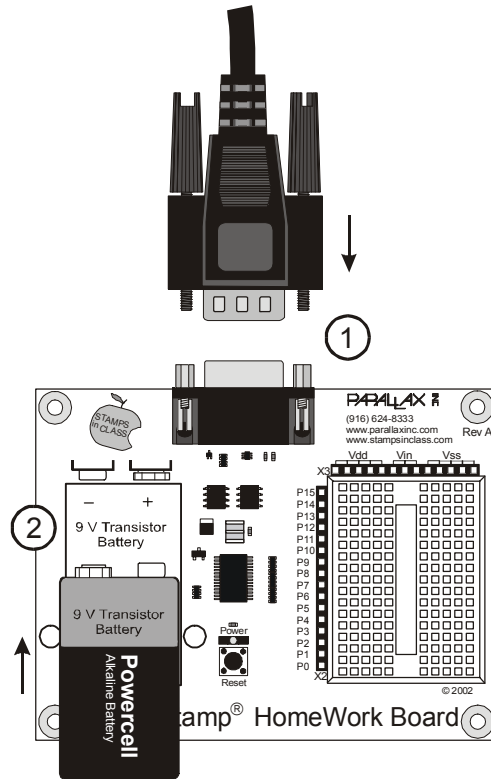



Figura 1-31
Scheda HomeWork e cavo seriale

Collegate il cavo seriale e la pila 9 V alla scheda HomeWork.

Collaudo della Comunicazione

L' Editor del BASIC Stamp ha un'opzione per assicurarsi che il vostro PC o laptop possa comunicare con il BASIC Stamp.

√ Doppio-click sull'icona dell'Editor del BASIC Stamp sul desktop del vostro computer. Dovrebbe apparire simile a quello illustrato Figura 1-32.

Figura 1-32
Icona dell'Editor del BASIC Stamp

Cercare un icona simile a questa sul desktop del vostro computer.

Il Menù di Avvio di Windows può anche essere usato per attivare l'Editor del BASIC Stamp. Clickare il bottone "Avvio" di Windows, quindi selezionate *Programs → Parallax, Inc. → Stamp Editor 2…*, poi clickare l'icona dell'*Editor del BASIC Stamp*.

La finestra dell'Editor del BASIC Stamp deve essere similare a quella mostrata in Figura 1-33.

La prima volta che usate l'Editor del BASIC Stamp, può mostrare alcuni messaggi ed una lista di porte seriali identificate dal programma.

√ Per assicurarsi che il BASIC Stamp sta comunicando con il computer, clickare il menù *Run*, quindi selezionare *Identify*.

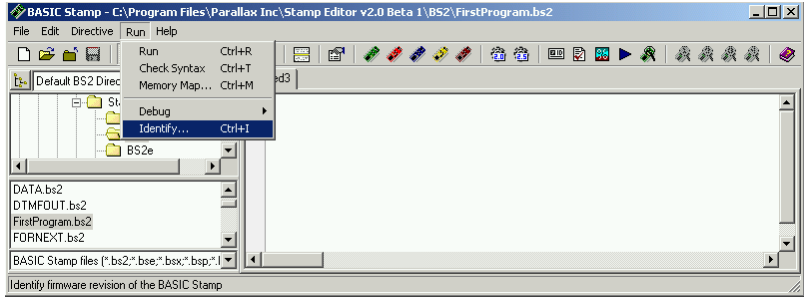



Figura 1-33
Editor del BASIC Stamp

Dal menu Run Selezionare Identify.

Apparirà una finestra di identificazione simile a quella mostrata in Figura 1-34. L'esempio nella figura, mostra che un BASIC Stamp 2 è stato rilevato sulla porta COM2.

Controllare nella finestra di identificazione per assicurarsi che il BASIC Stamp 2 sia stato rilevato su una delle porte seriali. Se il BASIC Stamp 2 è stato rilevato, siete pronti per l'Esercizio #4: .

√ Se la finestra di identificazione non rileva un BASIC Stamp 2 su alcuna delle porte COM, andare all'Appendice E: .

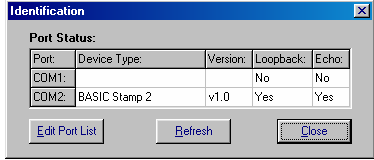



Figura 1-34
Finestra di Identificazione

Esempio: Il BASIC Stamp 2 rilevato sulla COM2.

ESERCIZIO #4: IL PRIMO PROGRAMMA

Il Primo Programma che scriverete e proverete, dirà al BASIC Stamp di inviare un messaggio al vostro PC o laptop. La Figura 1-35 come il BASIC Stamp invia una serie di uno e zero per comunicare i caratteri di testo visualizzati dal PC o laptop. Questi uno e zero sono chiamati numeri binari. L'Editor del BASIC Stamp ha la capacità di rilevare e visualizzare questi messaggi come potrete presto costatare.

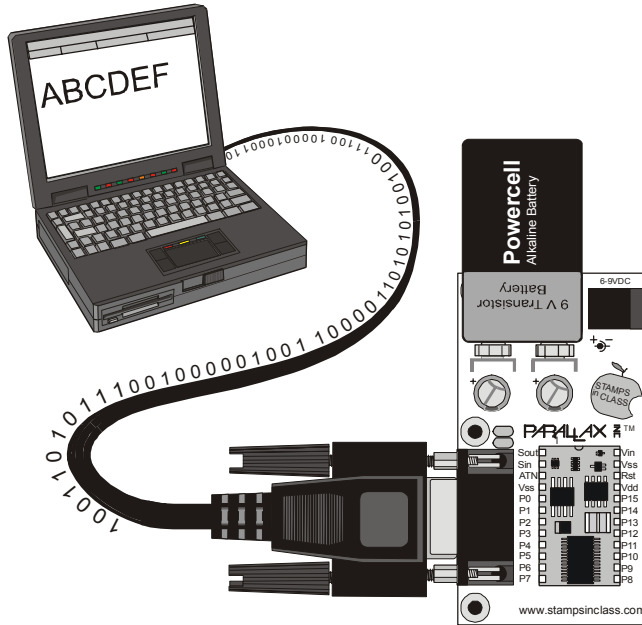



Figura 1-35
Messaggi dal BASIC Stamp al vostro Computer.

Il BASIC Stamp invia caratteri al vostro PC o laptop trasmettendo un flusso di uno e zero binari. L'Editor del BASIC Stamp può rilevare e convertire questi codici binari in caratteri e visualizzarli.

Il Primo Programma

Il Listato di programma che digiterete nell'Editor del BASIC Stamp e scaricherete nel BASIC Stamp sarà sempre visualizzato con un sottofondo grigio. Ecco un esempio:

Programma Esempio: FirstProgram.bs2

```
' Che cosa è un Microcontrollore- FirstProgram.bs2
' BASIC Stamp invia messaggi al Terminale di Debug.

' {$STAMP BS2}
```

```
' {$PBASIC 2.5}

DEBUG "Hello, it's me, your BASIC Stamp!"

END
```

√ Digitare questo programma nell'Editor del BASIC Stamp come mostrato in Figura 1-36.

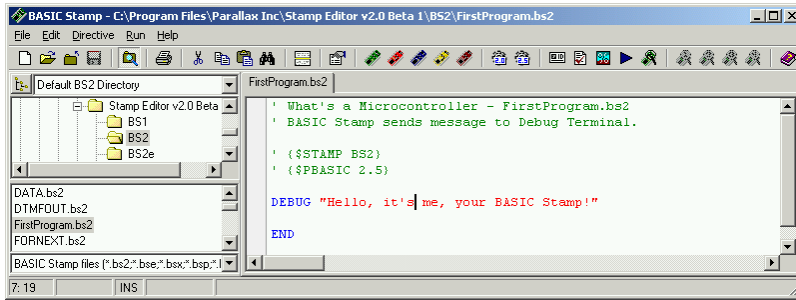



Figura 1-36
First Program digitato nell'Editor

Digitate il vostro primo programma nell'Editor del BASIC Stamp come mostrato qui.

√ Salvate il vostro lavoro clickando *File* e selezionare *Salva*, (mostrato in Figura 1-37).

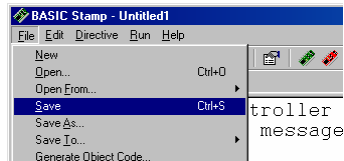



Figura 1-37
Salvare il primo programma

√ Digitare il nome FirstProgram nel campo *File name* in basso nella finestra *Salva come* similmente alla Figura 1-38.

√ Clickare il bottone *Save*.

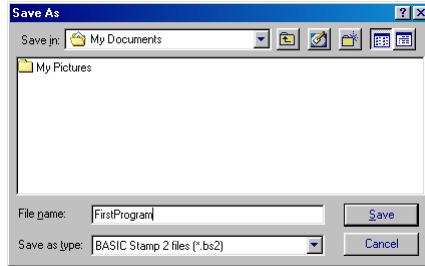



Figura 1-38
Inserire il nome del file

La prossima volta che salvate, l'Editor del BASIC Stamp salverà automaticamente con lo stesso nome (FirstProgram.bs2) a meno che gli direte di salvare con un nome differente clickando *File* e selezionando *Save As* (invece di *Save*).

√ Clickare *Run*, e selezionate *Run* dal menu che appare (clickandolo) come mostrato in Figura 1-39.

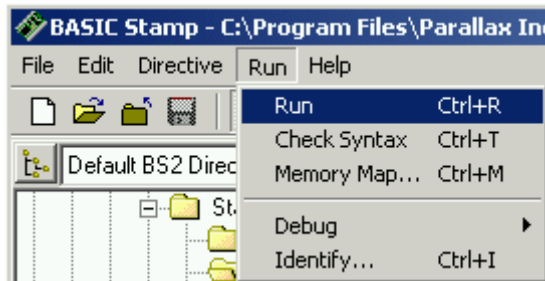



Figura 1-39
Attivare il vostro Primo Programma

Una finestra di progresso del Download apparirà brevemente per indicare che il programma viene trasmesso dal PC o LapTop al vostro BASIC Stamp. La Figura 1-40 mostra il Terminale di Debug che dovrebbe apparire quando il download è completato. Potete provare a voi stessi che questo è un messaggio dal BASIC Stamp premendo e rilasciando il tasto marcato *Rst* (Board of Education) oppure *Reset* (HomeWork Board). Questo tasto è chiamato il tasto di reset. Ogni volta che lo premete e rilasciate, il programma ripartirà da capo, e vedrete un'altra copia del messaggio visualizzato nella finestra del terminale di Debug.

√ Premere e rilasciare il tasto reset. Vedete un altro messaggio "Hello…" apparire nella finestra del Terminale di Debug?

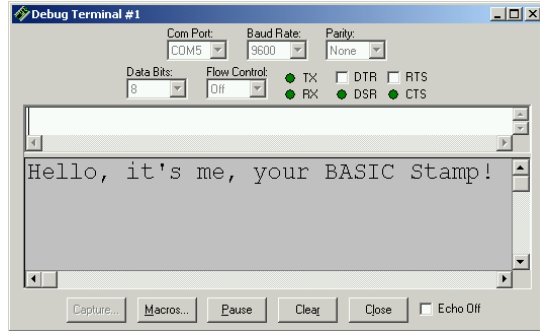



Figura 1-40
Finestra del Terminale di Debug

La Finestra del Terminale di Debug, visualizza i messaggi inviati al PC/laptop dal BASIC Stamp.

L'Editor del BASIC Stamp possiede tasti scorciatoia per i compiti più comuni. Per esempio, per attivare un programma, potete premere insieme i tasti 'Ctrl' e 'R'. Potete anche clickare il bottone *Run*. È il triangolo blù mostrato in Figura 1-41 che assomiglia al tasto Play di un CD player. Se puntate con il mouse sul bottone *Run* sarà mostrato che cosa fa quel tasto (suggerisce *Run*). Potete ottenere suggerimenti simili per sapere cosa fanno altri bottoni, puntandoli con il mouse.

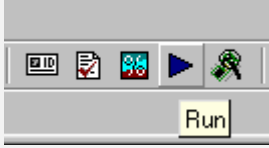



Figura 1-41

Tasti scorciatoia dell'Editor del BASIC Stamp

Come funziona FirstProgram.bs2

Nell'esempio, le prime due linee sono chiamate commenti. Un commento è una linea di testo che viene ignorata dall'Editor del BASIC Stamp, perché è un aiuto agli umani per comprendere il programma, e non per il BASIC Stamp. In PBASIC, qualsiasi cosa alla destra di un apostrofo, viene normalmente considerato essere un commento dall'Editor del BASIC Stamp. Il primo commento dichiara da quale libro proviene il programma esempio, e quale è il nome del programma. Il secondo commento contiene una descrizione stringata, di una riga, che spiega cosa fa il programma.

```
' Che cosa è un Microcontrollore- FirstProgram.bs2
' Il BASIC Stamp invia messaggi al Terminale di Debug.
```

Sebbene i commenti vengano di massima ignorati, l'Editor del BASIC Stamp cerca nei commenti se ci sono direttive speciali. Ogni programma in questo testo, userà queste due direttive:

```
' {$STAMP BS2}
```

```
' {$PBASIC 2.5}
```

La prima direttiva è chiamata Direttiva Stamp, dice all'Editor del BASIC Stamp che programmerete un BASIC Stamp 2. La seconda direttiva è chiamata la Direttiva PBASIC, e dice all'Editor del BASIC Stamp che state usando la versione 2.5 del linguaggio di programmazione PBASIC.

Un comando è una parola che dice al BASIC Stamp di fare una determinata operazione. I primi due comandi in questo programma sono chiamati comandi **DEBUG**:

```
DEBUG "Hello, it's me, your BASIC Stamp!"
```

Questo commando dice al BASIC Stamp di inviare un messaggio al PC tramite il cavo seriale.

Il secondo comando si chiama **END**:

```
END
```

Questo commando è utile perché pone il BASIC Stamp, quando viene dato, in modalità a basso consumo. In modalità a basso consumo, il BASIC Stamp aspetta la pressione (e successivo rilascio) del tasto reset, o l'invio di un nuovo programma dall'Editor del BASIC Stamp. Se viene premuto sulla vostra scheda il tasto reset, il BASIC Stamp reinizierà il programma già presente. Se viene inviato un nuovo programma, verrà cancellato il vecchio, ed il nuovo programma sarà attivato.

Il Vostro Turno – Formattatori DEBUG e Caratteri di Controllo.

Un formattatore D**EBUG** è una parola codice che potete usare per fare apparire in un certo modo il messaggio che il BASIC Stamp invia al Terminale di Debug. **DEC** è un esempio di formattatore che fa visualizzare nel Terminale di Debug un valore decimale. Un esempio di carattere di controllo è **CR**, che viene usato per inviare un "a capo" al Terminale di Debug. Il testo o numero che viene dopo un **CR** apparirà sulla linea successiva. Potete modificare il vostro programma in modo che contenga più comandi **DEBUG** insieme con alcuni formattatori e caratteri di controllo. Di seguito viene indicato un esempio di come farlo:

√ Primo, salvate il programma con un nuovo nome clickando *File* e selezionando *Salva Come*.
√ Un nome nuovo per il file potrebbe essere FirstProgramYourTurn.bs2
√ Modificare i commenti all'inizio del programma cosicché si legga:

```
' Che cosa è un Microcontrollore- FirstProgramYourTurn.bs2
' BASIC Stamp invia messaggi al Terminale di Debug.
```

√ Aggiungere queste tre linee dopo il comando **DEBUG** ed il comando **END**:

```
DEBUG CR, "What's 7 X 11?"
DEBUG CR, "The answer is: "
DEBUG DEC 7 * 11
```

√ Salvate i cambiamenti che avete fatto clickando *File* e selezionando *Salva*.

Il vostro programma dovrebbe ora essere simile a quello mostrato in Figura 1-42.

√ Attivare il vostro programma modificato. Suggerimento: Dovrete clickare di nuovo *Run* dal menu Run, come in Figura 1-39 oppure clickare il tasto *Run*, come in Figura 1-41.

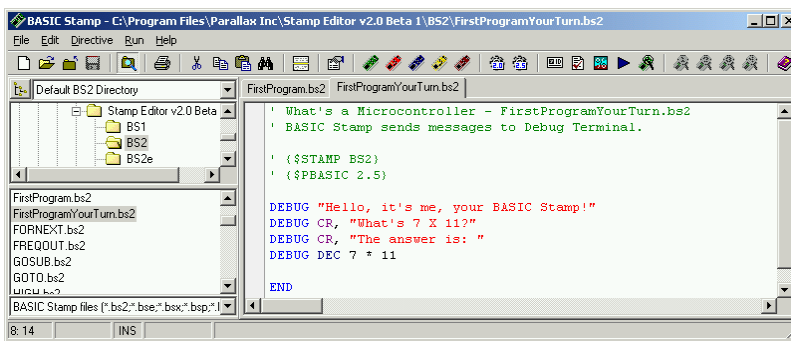



Figura 1-42
FirstProgram Modificato

Controllate il vostro lavoro con il programma esempio mostrato a fianco.

Dove è andato a finire il mio Terminale di Debug? Qualche volta il Terminale di Debug viene nascosto dietro la finestra dell'Editor del BASIC Stamp. Potrete riportarlo in primo piano usando il menu *Run* come mostrato a sinistra nella Figura 1-43, usando il tasto scorciatoia mostrato a destra nella figura, oppure il tasto F12 della vostra tastiera.

Figura 1-43

Terminale di Debug 1 in primo piano

Usando il menu (sinistra) oppure il tasto scorciatoia (destra).

Il vostro Terminale di Debug dovrebbe ora somigliare alla Figura 1-44.

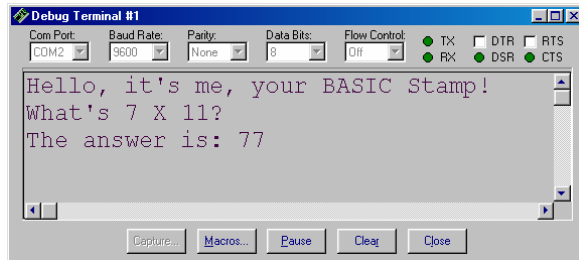



Figura 1-44
FirstProgram.bs2 Modificato. Risultato sul Terminale di Debug.

Assicuratevi di ottenere il risultato sperato quando farete nuovamente girare il vostro programma.

ESERCIZIO #5: ALLA RICERCA DI RISPOSTE

Il programma esempio che avete appena terminato, ha introdotto due comandi PBASIC: **DEBUG** ed **END**. Potrete trovare altre informazioni circa questi comandi e su come sono usati cercandoli, nell'aiuto dell'Editor del BASIC Stamp o nel Manuale del BASIC Stamp. Questo Esercizio vi guida con un esempio nella ricerca di informazioni su **DEBUG** usando l'Aiuto dell'Editor del BASIC Stamp ed il Manuale del BASIC Stamp.

Usare l'Aiuto dell'Editor del BASIC Stamp

√ Nell'Editor del BASIC Stamp, Clickare *Help*, quindi selezionare *Index* come mostrato in Figura 1-45.

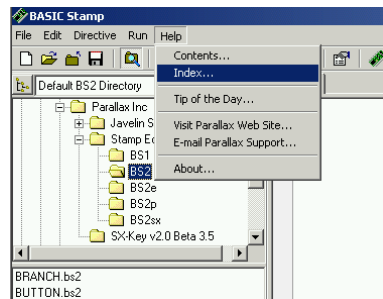



Figura 1-45
Selezionare Index dal Menu Aiuto

√ Digitare DEBUG nel campo etichettato *Type in the keyword to find:* (mostrato in Figura 1-46).

√ Quando nella lista sottostante appare la parola DEBUG, clickatela, quindi clickate il bottone *Display*.

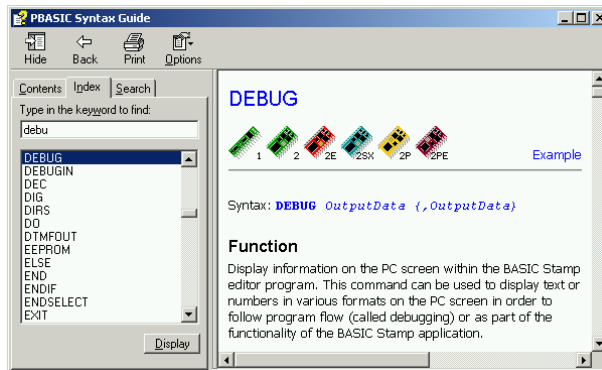



Figura 1-46
Cercare il commando DEBUG Usando l'Aiuto.

IL Vostro Turno

- √ Usare i cursori per trovare le spiegazioni del comando **DEBUG**. Notare che ci sono molte spiegazioni ed esempi che potete sperimentare.
- √ Clickare la barra *Contents*, e trovate DEBUG.
- √ Clickare la barra *Search*, e fare una ricerca per la parola DEBUG.
- √ Ripetere questa procedura per il comando **END**.

Procurarsi ed usare il Manuale del BASIC Stamp.

Il Manuale del BASIC Stamp è disponibile gratuitamente scaricandolo dal sito web della Parallax. E' anche incluso nel CD della Parallax. Può anche essere acquistato sotto forma di manuale stampato e rilegato.

Scaricare il Manuale BASIC Stamp dal sito web della Parallax

Usando un browser web, andare a www.parallax.com.

√ Puntare al menu *Downloads* per visualizzare le opzioni.
√ Puntare al link *Documentation* e clickarlo per selezionarlo.
√ Quando siete nella pagina della documentazione del BASIC Stamp, trovare il Manuale del BASIC Stamp.
√ Clickare sull'icona *Download* simile ad una cartella alla destra della scritta: "BASIC Stamp User's Manual Version 2.0 (3.2 MB)".

Visualizzare il Manuale del BASIC Stamp sul CD della Parallax

√ Clickare il link *Documentation*.
√ Clickare il + vicino alla cartella BASIC Stamps.
√ Clickare l'icona del Manuale del BASIC Stamp.
√ Clickare il bottone *View*.

√ La Figura 1-47 è un estratto dal Manuale del *Manuale del BASIC Stamp* v2.0 alla sezione Contenuti. Mostra che le informazioni sul comando **DEBUG** possono essere trovate a pagina 97.

Figura 1-47 Trovare il Comando DEBUG nell'Indice

La Figura 1-48 è un estratto dalla pagina 97 nel Manuale del BASIC Stamp v2. 0. il commando è spiegato in dettaglio insieme con esempi di programmi per dimostrare come può essere usato il comando **DEBUG**.

√ Un breve sguardo sulla spiegazione del comando **DEBUG** nel Manuale del BASIC Stamp.

√ Contare il numero degli esempi di programmi nella sezione **DEBUG**. Quanti ce ne sono?

5: BASIC Stamp Command Reference - DEBUG

DEBUG | BS1 | BS2 | BS2e | BS2sx | BS2p |

DEBUG *OutputData {, OutputData}*

Function

Display information on the PC screen within the BASIC Stamp editor program. This command can be used to display text or numbers in various formats on the PC screen in order to follow program flow (called debugging) or as part of the functionality of the BASIC Stamp application.

Figura 1-48
Rivedere il comando **DEBUG** nel Manuale del BASIC Stamp.

IL Vostro Turno

√ Usare l'indice nel Manuale del BASIC Stamp per cercare il comando **DEBUG**.
√ Cercare il comando **END** nel Manuale del BASIC Stamp.

ESERCIZIO #6: QUANDO AVETE FINITO

E' importante spengere il vostro BASIC Stamp e la Board of Education (o la HomeWork Board) per diversi motivi. Primo, le vostre pile dureranno più a lungo se il vostro sistema non assorbe energia quando non lo usate. Secondo, negli esperimenti futuri, costruirete dei circuiti nella piastra prototipi della Board of Education o della HomeWork Board.

I circuiti prototipi non devono mai essere lasciati accesi e incustoditi. Non potete sapere che genere di problema può accadere mentre non ci siete.

Scollegare sempre l'alimentazione dalla vostra Board of Education o dalla HomeWork Board, anche se la abbandonate per un minuto o due.

Se siete in una classe, il vostro istruttore può avere ulteriori istruzioni, come scollegare il cavo seriale, riporre la vostra Board of Education o HomeWork Board in un posto sicuro, etc. A parte questi dettagli, la cosa più importante che dovrete sempre fare è scollegare l'alimentazione quando avete finito.

Scollegare l'Alimentazione

Con la Board of Education Rev C, scollegare l'alimentazione è facile :

√ Se state usando la Board of Education Rev C, posizionare l'interruttore a tre posizioni nella posizione 0 spingendolo a sinistra come mostrato in Figura 1-49.

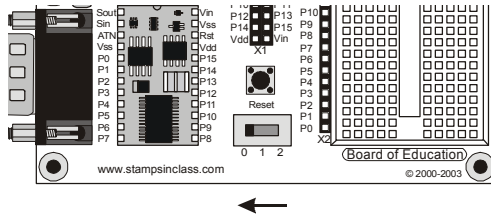



Figura 1-49
Spegnimento della

Board of Education Rev C

Non togliere il BASIC Stamp dal suo zoccolo nella Board of Education. Resistete alla tentazione di riporre la vostra Board of Education ed il BASIC Stamp separatamente. Ogni volta che il BASIC Stamp viene rimosso e reinserito nello zoccolo sulla Board of Education, si possono fare errori che lo danneggeranno. Sebbene il BASIC Stamp possa essere rimosso da uno zoccolo all'altro durante un progetto più grande, ciò non sarà necessario durante gli esercizi di questo testo.

Anche lo spegnimento della scheda BASIC Stamp HomeWork è facile:

√ Se state usando la scheda BASIC Stamp HomeWork, scollegare la pila come mostrato in Figura 1-50.

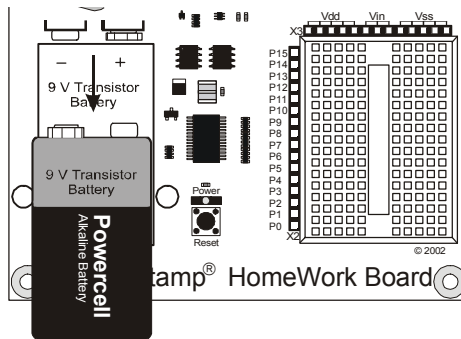



Figura 1-50
Scollegare la pila dalla scheda HomeWork

Anche la Board of Education Rev B deve essere spenta, sia rimuovendo la pila che scollegando lo spinotto di alimentazione.

IL Vostro Turno

√ Adesso spengete la vostra scheda.

SOMMARIO

Questo capitolo vi ha guidato attraverso le cose seguenti:

- Un'introduzione ad alcuni dispositivi che contengono microcontrollori.
- Ulna introduzione al BASIC Stamp.
- Una rivista di alcune interessanti invenzioni che sono state fatte con BASIC Stamp.
- Dove trovare il software gratuito dell'Editor del BASIC Stamp che userete praticamente in tutti gli esperimenti di questo testo.
- Come installare l'Editor del BASIC Stamp.
- Un'introduzione al BASIC Stamp, alla Board of Education, ed alla scheda HomeWork.
- Come approntare l'hardware del vostro BASIC Stamp.
- Come collaudare il vostro software e hardware.
- Come scrivere ed avviare un programma PBASIC.
- Usare i comandi **DEBUG** ed **END.**
- Usare i caratteri di controllo ed il formattatore **DEC.**
- Come usare l'Aiuto dell'Editor del BASIC Stamp ed il Manuale del BASIC Stamp.
- Come spengere la vostra Board of Education o la vostra scheda HomeWork quando avete finito.

Domande

1. Che cos'è un microcontrollore?
2. Il BASIC Stamp è un microcontrollore, o ne contiene uno?
3. Quali caratteristiche controllereste per sapere se un oggetto come una radio sveglia o un telefono cellulare contiene un microcontrollore?
4. Qual è lo scopo del cavo seriale?
5. Quando il BASIC Stamp invia un carattere al vostro PC/laptop, che numeri sono usati per inviare il messaggio tramite il cavo seriale?
6. Che cosa dovreste fare dopo aver digitato il vostro programma nell'Editor del BASIC Stamp per farlo girare sul BASIC Stamp?
7. Come si chiama la finestra che visualizza i messaggi inviati dal BASIC Stamp al vostro PC/laptop?
8. Che cosa fa un apostrofo all'inizio di una linea di codice programma PBASIC?

9. Quali comandi avete imparato in questo capitolo?
10. Diciamo che vogliate fare una pausa dal vostro progetto BASIC Stamp per un caffè, o può darsi per una pausa più lunga e tornare al progetto dopo un paio di giorni. Cosa dovrete sempre fare prima di fare la vostra pausa?

Esercizi

1. Spiegare cosa potete fare con ciascun comando PBASIC che avete imparato in questo capitolo.
2. Spiegate che cosa accadrebbe se toglieste tutti i caratteri di controllo **CR** dai comandi **DEBUG** sotto riportati e scrivete come si vedrebbero nel Terminale di Debug.

```
DEBUG "Hello, it's me, your BASIC Stamp!"
DEBUG CR, "What's 7 X 11?"
DEBUG CR, "The answer is: "
```

3. Spiegare cosa fa l'asterisco in questo commando:

```
DEBUG DEC 7 * 11
```

4. Indovinare che cosa visualizzerebbe il Terminale di Debug se eseguiste questo comando:

```
DEBUG DEC 7 + 11
```

5. C'è un problema con questi due comandi. Quando fate partire il programma, i numeri sono visualizzati tutti attaccati cosicché sembrano un unico numero più grande invece che due piccoli. Modificate questi due comandi in modo che la risposta appaia su linee differenti nel Terminale di Debug.

```
DEBUG DEC 7 * 11
DEBUG DEC 7 + 11
```

Progetti

1. Usare **DEBUG** per visualizzare la soluzione al problema matematico: 1 + 2 + 3 + 4.
2. Usare la sezione a pagina 26 come modello per salvare il vostro file con un nuovo nome e modificatelo. Usare il nome di file ProgramCh01Project02.bs2. Aggiungete al programma questa linea, ed attivatelo:

```
DEBUG 65, 66, 67, 68, 69, 70
```

Quindi, inserite il formattatore **DEC** prima di ciascuno di questi numeri in modo che la linea si legga:

```
DEBUG DEC 65, DEC 66, DEC 67, DEC 68, DEC 69, DEC 70
```

Fate ripartire il programma e descrivete cosa fa il formattatore **DEC**. La maggior parte dei microcontrollori e dei computer PC aderiscono alla codifica standard americana per lo scambio di informazioni. Probabilmente lo vedrete citato come codice ASCII. Il codice ASCII della 'A' è 65, 'B' è 66, 'C' è 67, e così via. Il codice ASCII per un carattere spazio è 32, ed il codice per un a capo è 13. provare queste cose e spiegare in una relazione di un paragrafo che cosa fa ciascuna di queste linee.

```
DEBUG "Hello!"
DEBUG 32, 32, 32, 32
DEBUG "Hello again!"
DEBUG 13
DEBUG "Goodbye."
```

3. Predire che cosa vi aspettate di vedere se rimuovete il formattatore **DEC** da questo comando. Usare un programma PBASIC per verificare la vostra predizione.

```
DEBUG DEC 7 * 11
```

4. Vedere ancora una volta nella Figura 1-35 a pagina 22. Come potete inviare il numero 65 (che rappresenta una 'A') se siete costretti agli zero ed uno? La risposta è che il numero 65 può essere rappresentato usando uno e zero. Apprenderete di più circa il sistema binario (base-2) in seguito. Per ora, semplicemente modificate un programma aggiungendogli questo segmento di codice e verificare che faccia la stessa cosa del segmento di codice nel progetto 2.

```
' invia i codici ASCII 65, 66, 67, 68, 69, and 70.
' il Terminale di Debug visualizzerà "ABCDEF".

DEBUG %01000001, %01000010, %01000011
DEBUG %01000100, %01000101, %01000110
```

5. Quali linee potrete cancellare nel programma FirstProgramYourTurn.bs2 se mettete il comando sotto riportato nella linea subito prima del comando **END**? Verificate la vostra ipotesi (la vostra predizione di cosa avverrà). Assicuratevi di salvare FirstProgramYourTurn.bs2 con il nuovo nome FirstProgramCh01Project05.bs2. quindi fate le vostre modifiche, salvare ed attivate il vostro programma.

```
DEBUG "What's 7 X 11?", CR, "The answer is: ", DEC 7 * 11
```

Ulteriori indagini

In questo capitolo, avete visitato la sezione software del sito web della Parallax oppure il CD Parallax per ottenere una copia dell'Editor del BASIC Stamp. Potete andare alla sezione documentazione sia del sito web della Parallax, che, del CD Parallax per avere una copia gratuita di questo testo e del *Manuale del BASIC Stamp*. Copie stampate possono anche essere acquistate presso la Parallax.

"*Manuale del BASIC Stamp*", Manuale Utente, Version 2.0c, Parallax Inc., 2000

Potete apprendere molto altro circa i comandi **`DEBUG`** ed **`END`** cercandoli nel *Manuale del BASIC Stamp*. Li potete trovare usando l'Indice. Il *Manuale del BASIC Stamp* ha molti altri esempi che potete provare, insieme con lezioni simili a quelle nel progetto che avete appena completato.

Capitolo #2: Luci accese - Luci spente.

2

INDICATORI LUMINOSI

Gli Indicatori Luminosi sono così comuni che molte persone tendono a non farvi troppo caso. La Figura 2-1 mostra tre indicatori luminosi su una stampante laser. Secondo quale luce è accesa, chi usa la stampante sa se la stampante sta operando correttamente oppure No. Sono elencati di seguito alcuni esempi di dispositivi con indicatori luminosi: autoradio, videoregistratori, giradischi, stampanti e pannelli di controllo di impianti di allarme.

Figura 2-1
Indicatori
Luminosi

Accendere e spengere un indicatore luminoso è una semplice faccenda di connetterlo e disconnetterlo da una sorgente di alimentazione. In alcuni casi, l'indicatore luminoso, e collegato direttamente alla pila o all'alimentatore, come per esempio l'indicatore luminoso della Board of Education. Altri indicatori luminosi sono accesi e spenti tramite un microcontrollore interno al dispositivo. Questi sono solitamente indicatori di stato che vi indicano che cosa sta facendo il dispositivo.

FAR EMETTERE LUCE AD UN DIODO EMETTITORE DI LUCE (LED)

La maggioranza degli indicatori luminosi che vedete nei dispositivi sono chiamati diodi emettitori di luce. Vedrete spesso un diodo emettitore di luce indicato nei libri e sugli schemi con le lettere LED. Il nome viene di solito pronunciato come tre lettere separate:

"L-E-D". Potete costruire un circuito LED ed alimentarlo, ed il LED emetterà luce. Se lo scollegate, il LED si spengerà.

Un circuito LED può essere collegato al BASIC Stamp, ed il BASIC Stamp può essere programmato per collegare e scollegare l'alimentazione del circuito LED. Questo è molto più semplice che cambiare manualmente i collegamenti del circuito o collegare e scollegare la pila. Il BASIC Stamp può inoltre essere programmato per fare le seguenti cose:

- Accendere e spengere un circuito LED ad intervalli differenti
- Accendere e spengere un circuito LED un determinato numero di volte.
- Controllare più di un LED
- Controllare il colore di un LED bicolore (a due colori).

ESERCIZIO #1: COSTRUIRE E COLLAUDARE IL CIRCUITO LED

É importante collaudare singolarmente i componenti prima di inserirli in un sistema più grande. Questo Esercizio mette a fuoco la costruzione ed il collaudo di due circuiti diversi con i LED. Il primo circuito fa accendere un LED. Il secondo circuito non lo fa accendere. Nell'Esercizio che viene dopo di questo, inserirete un circuito LED in un sistema più grande, collegandolo al BASIC Stamp. Scriverete quindi dei programmi che faranno accendere il LED dal BASIC Stamp, quindi lo faranno spengere. Verificando prima che ciascun circuito LED funzioni, potrete essere più sicuri che funzionerà quando lo collegherete al BASIC Stamp.

Conosciamo la Resistenza

Una resistenza è un dispositivo che si oppone al fluire dell'elettricità. Questo flusso di elettricità è chiamato corrente. Ogni resistenza ha un valore che indica con quanta forza resiste al flusso di corrente. Questo valore della resistenza viene indicato in ohm ed il simbolo per gli ohm è la lettera dell'alfabeto Greco - Ω. La resistenza con cui lavorerete in quest'esercizio, è la resistenza da 470 Ω mostrata nella Figura 2-2. La resistenza ha due fili chiamati terminali, che escono da ciascun'estremità. Tra i due terminali c'è un contenitore ceramico, ed è questa la parte che oppone resistenza al flusso della corrente. La maggior parte degli schemi che usano il simbolo della resistenza a sinistra, una linea a zig zag, indicano che la persona che dovrà costruire il circuito, deve usare una resistenza da 470 Ω. E' chiamato simbolo schematico. Il disegno sulla destra, è parte di uno schema usato in alcuni testi per allievi alle prime armi del Corso Stamps in Class per aiutarli ad individuare la resistenza nel kit.

2

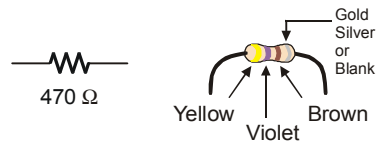



Figura 2-2
Disegno di una resistenza da 470 Ω

Simbolo Schematico (a sinistra) e disegno del componente (a destra).

Le resistenze come quelle che stiamo usando in quest'Esercizio, hanno fasce colorate che indicano il valore di resistenza. Ciascun valore di resistenza ha una combinazione diversa di colori. Per esempio il codice colori per la resistenza da 470 Ω è giallo, viola, marrone.

Ci può essere una quarta fascia che indica la tolleranza della resistenza. La tolleranza è misurata in percentuale, ed indica di quanto il valore reale si può discostare dal valore nominale. La quarta fascia può essere oro (5%), argento (10%), o essere assente (20%). Per gli esercizi di questo testo, la tolleranza della resistenza non è importante, ma lo è il valore.

Ciascuna fascia colorata indica una cifra del valore della resistenza e questi valori/cifre sono elencati nella Tabella 2-1. Figura 2-3.

Tabella 2-1: Valori del codice colori delle resistenze

Digit	Color
0	Nero
1	Marrone
2	Rosso
3	Arancio
4	Giallo
5	Verde
6	Blu
7	Viola
8	Grigio
9	Bianco

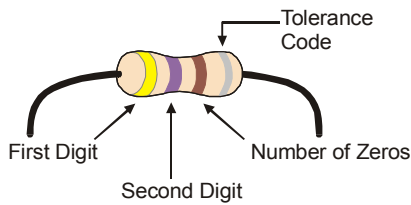



Figura 2-3
Codice colori delle resistenze

Di seguito c'è un esempio che mostra come usare la Tabella 2-1 e la Figura 2-3 per sapere il valore di una resistenza e che vi prova che giallo-viola-marrone è una resistenza da 470 Ω:

- La prima fascia è gialla, significa che la prima cifra è 4.
- La seconda fascia è viola, significa che la seconda cifra è 7.
- La terza fascia è marrone, dal momento che marrone è 1, significa aggiungere 1 zero alle prime due cifre.

Giallo-Viola-Marrone = 4-7-0.

Conosciamo il LED.

Un diodo è un dispositivo che fa scorrere la corrente solo in un verso, un diodo emettitore di luce (LED) emette luce quando la corrente lo attraversa. Al contrario del codice colori delle resistenze, il colore di un LED normalmente indica di quale colore s'illuminerà quando la corrente lo attraversa. La polarità di un LED è contenuta nella sua forma. Dal momento che un LED è un dispositivo polarizzato, vi dovete assicurare di collegarlo nella giusta maniera, o non funzionerà come voluto.

La Figura 2-4 mostra lo schema di collegamento di un LED e lo schema pratico. Un LED ha due terminali. Uno è chiamato anodo, l'altro è chiamato catodo. In quest'Esercizio, dovrete collegare il LED ad un circuito, e dovrete fare attenzione ed assicurarvi che l'anodo ed il catodo siano collegati al circuito correttamente. Sullo schema pratico, l'anodo è marcato con il simbolo più (+). Sullo schema elettrico, l'anodo è la parte larga del triangolo. Sullo schema pratico il catodo è il terminale non contrassegnato, e sullo schema elettrico,il catodo è la linea al vertice del triangolo.

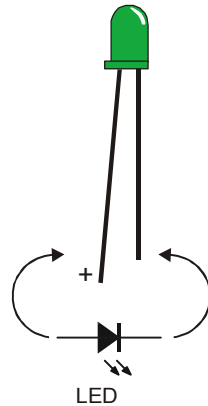



Figura 2-4
Schema pratico del LED e simbolo elettrico.

Schema pratico (sopra)
Simbolo elettrico (sotto).

Lo schema pratico del LED nelle figure successive avrà il positivo vicino al terminale anodo.

Quando iniziate l'assemblaggio del vostro circuito, assicuratevi di controllare e confrontare lo schema elettrico con lo schema pratico. Sullo schema pratico, notare che i terminali del LED sono di lunghezza diversa. Il terminale più lungo è l'anodo del LED, il terminale corto è il catodo. Inoltre se guardate attentamente il corpo plastico del LED è quasi rotondo con un piccola zona piatta vicino al terminale corto identificando così il catodo. Questo fatto è particolarmente utile se i terminali sono stati tagliati alla stessa lunghezza.

Elenco componenti del circuito LED

(1) LED – Verde
(1) Resistenza – 470 Ω (giallo-viola-marrone)

Identificazione dei componenti: In aggiunta allo schema pratico della Figura 2-2 e della Figura 2-4, per aiutarvi ad identificare i componenti del kit che vi servono, potete guardare le fotografie sull'ultima pagina del libro, ciò vale per questo e per tutti gli altri esperimenti. Per ulteriori informazioni sui componenti di queste foto, vedere l'Appendice B: Equipaggiamenti ed Elenco Componenti.

Costruzione del circuito di prova del LED

Costruirete il circuito inserendo i terminali del LED e della resistenza nei fori della piastra per prototipi come mostrato nella Figura 2-5. In questa piastra per prototipi ci sono dei connettori neri lungo il fianco sinistro e sul lato superiore. I connettori neri sul lato superiore, sono etichettati: Vdd, Vin, e Vss. Queste etichette si chiamano terminali di alimentazione, e saranno usati per fornire elettricità ai vostri circuiti. I connettori neri sul lato sinistro sono etichettati con le sigle P0, P1, fino a P15. Queste sono le connessioni che userete per collegare il vostro circuito ai piedini di ingresso/uscita del BASIC Stamp. La zona bianca con molti fori, si chiama area prototipi senza saldature. La userete per collegare tra loro i componenti dei vostri circuiti.

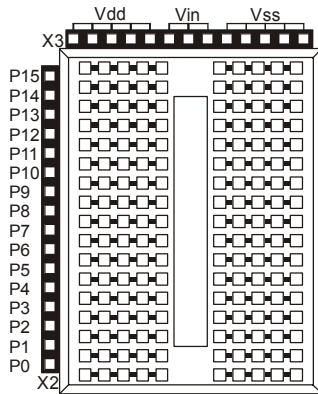



Figura 2-5
Piastra per prototipi

Connettori di alimentazione (fori neri sul lato superiore), Piedini di ingresso/uscita (connettori neri sul lato sinistro), area prototipi senza saldature (connettori bianchi)

Piedini di ingresso/uscita sono di solito chiamati pin di I/O, e dopo aver collegato il vostro circuito ad uno o più di questi pin di I/O, potete programmare il vostro BASIC Stamp per controllare il circuito (ingressi) o inviare segnali al circuito (uscite). Proverete tutto questo nel prossimo esercizio.

La Figura 2-6 mostra una coppia di circuiti assemblati nell'area prototipi. L'area prototipi, è divisa in file di cinque connettori collegati insieme. Ciascuna fila di cinque connettori può essere usata per collegare fino a cinque fili o terminali di componenti fra di loro. Gli esempi di terminali di componenti e fili che sono connessi fra di loro,sono cerchiati. Sebbene non siano cerchiati, probabilmente direte che un terminale di una resistenza è collegato a P7 ed un altro è collegato a P15. Uno dei terminali dei LED è collegato a Vss, ed un filo collegato ad uno dei circuiti è anche lui collegato a Vss.

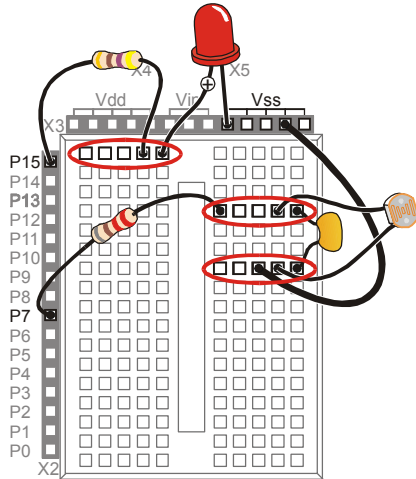



Figura 2-6
Circuito esempio della scheda prototipi.

NON COSTRUIRE QUESTO CIRCUITO.

Le file di 5 connettori sull'area prototipi sono usate per collegare I terminali di due o più componenti e sono cerchiate. I terminali di alcuni componenti, sono collegati a P7, P15, e Vss.

Siete ora pronti per costruire il circuito mostrato in Figura 2-7 (sotto) inserendo la resistenza ed il LED nei connettori dell'area prototipi. Seguite i seguenti passi:

√ Togliere l'alimentazione dalla vostra Board of Education o dalla vostra HomeWork Board.
√ Usare la Figura 2-4 per decidere quale terminale è collegato al catodo del LED. Cercare il terminale più corto e la zona piatta sul corpo plastico del LED.
√ Inserire il catodo del LED in uno dei connettori neri etichettati Vss sul lato superiore della scheda prototipi.
√ Inserire l'anodo del LED (l'altro terminale, il più lungo) nel connettore della porzione di area prototipi mostrata.
√ Inserire uno dei terminali della resistenza nella stessa fila di connettori dell'anodo del LED. Questo collegherà insieme i due componenti.
√ Inserire l'altro terminale della resistenza in uno dei connettori etichettati Vdd.

La polarità non interessa nelle resistenze ma è importante per il LED. Se collegate il LED al contrario, il LED non emetterà luce quando alimenterete il circuito. La resistenza si oppone al passaggio di corrente. La resistenza non ha polo positivo o negativo.

√ Rialimentate la vostra Board of Education o la vostra HomeWork Board.
√ Controllate che il LED verde si accenda. Dovrebbe illuminarsi di verde.

2

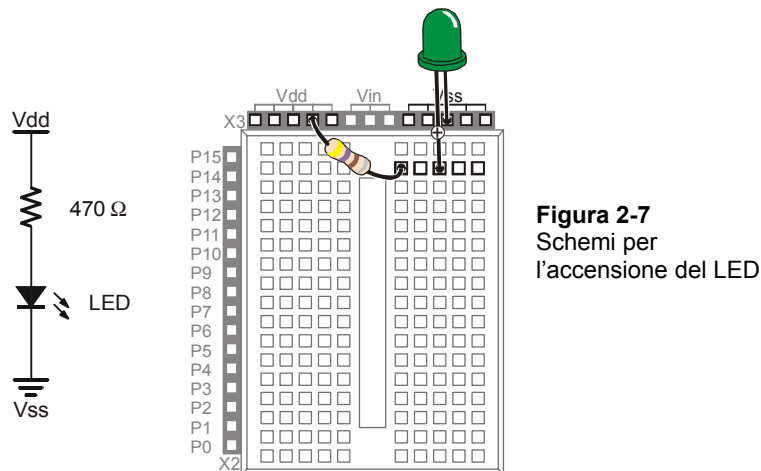



Figura 2-7
Schemi per l'accensione del LED

Se il LED verde non si accende quando alimentate la scheda:

√ Alcuni LED sono più luminosi se lì guardate da sopra. Provate a guardare direttamente sulla cupoletta del contenitore plastico.
√ Se la stanza è fortemente illuminata provate a spengere l'illuminazione oppure usate le mani per mettere in ombra il LED.

Se ancora non vedete nessuna luce verde, fare questi tentativi:

√ Ricontrollate per assicurarvi che l'anodo ed il catodo siano collegati correttamente. Se questo non è, semplicemente rimuovete il LED e reinseritelo al contrario. Non danneggerete il LED se lo collegherete al contrario, semplicemente non si accenderà. Quando sarà collegato per il giusto verso, si accenderà.
√ Ricontrollate per assicurarvi che avete assemblato il circuito esattamente come mostrato in Figura 2-7.
√ Se state usando un kit "Che cosa è un Microcontrollore" e qualcuno lo ha usato prima di voi provate a cambiare il LED nel caso che sia difettoso.
√ Se siete in laboratorio, controllate insieme all'insegnante.

Ancora bloccati? Se non avete un insegnante o un amico che vi può aiutare, potete sempre controllare tramite il gruppo di discussione dello Stamps in Class. Nella prima pagina di questo libro ci sono gli indirizzi internet dove trovare il gruppo di discussione dello Stamps in Class. Se il gruppo è incapace di risolvere il problema, potete contattate il supporto tecnico della Parallax seguendo il collegamento Support nel sito www.parallax.com.

Come funziona il circuito di prova del LED.

I terminali Vdd e Vss forniscono 'pressione' elettrica allo stesso modo di come farebbe una pila. I connettori Vdd equivalgono al polo positivo della pila, ed i connettori Vss sono uguali al polo negativo. La Figura 2-8 mostra come applicando 'pressione' elettrica ad un circuito con una pila, costringe gli elettroni a fluire attraverso di essa. Questo flusso di elettroni si chiama corrente elettrica, o più generalmente corrente. La corrente elettrica è limitata dalla resistenza. Questa corrente è la causa dell'illuminazione del diodo.

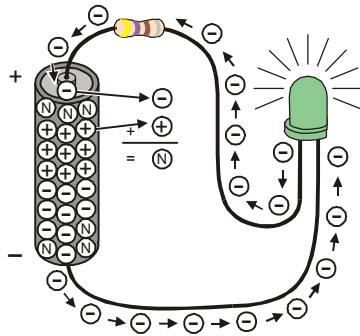



Figura 2-8
Flusso di elettroni nel circuito LED acceso.

I segni meno nei cerchietti vengono usati per mostrare gli elettroni che scorrono dal polo negativo della pila al suo polo positivo.

Le Reazioni chimiche all'interno della pila forniscono corrente al circuito. Il polo negativo della pila contiene un composto di molecole con più elettroni del normale (rappresentate nella Figura 2-8 dai segni meno). Il polo positivo della pila è fatto di un composto chimico con molecole a cui mancano elettroni (rappresentate dai segni più). Quando un elettrone lascia una molecola per scorrere attraverso il filo, viene chiamato elettrone libero (anche lui rappresentato da un segno meno). La molecola che ha perso l'elettrone, non ha più una carica negativa; è ora chiamata neutra (indicata da una N). Quando un elettrone giunge al polo positivo, si unisce ad una molecola a cui mancava un elettrone, cosicché quella molecola diventa neutra.

La Figura 2-9 mostra come il flusso di corrente attraverso il circuito LED venga descritto usando la simbologia schematica. La pressione elettrica attraverso il circuito si chiama

tensione. I segni + e – sono usati per mostrare dove la tensione viene applicata al circuito. La freccia mostra il verso della corrente nel circuito. Questa freccia punta di solito nel verso opposto al flusso reale degli elettroni. Si ritiene che Benjamin Franklin non si fosse reso conto degli elettroni quando decise di rappresentare il flusso di corrente come cariche passanti dal polo positivo al polo negativo di un circuito. Al momento in cui i fisici scoprirono la vera natura della corrente elettrica, tale convenzione era già notevolmente diffusa, per cui è rimasta.

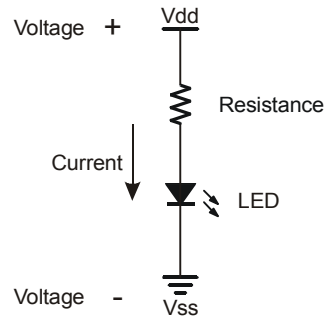



Figura 2-9
Schema di un circuito LED acceso che mostra la tensione ed il verso convenzionale del flusso della corrente

I segni + e – mostrano la tensione applicata al circuito, e la freccia mostra il verso di scorrimento della corrente attraverso il circuito.

Uno schema elettrico (come la Figura 2-9) è un disegno che rappresenta come uno o più circuiti sono collegati. Gli schemi sono usati da studenti, dilettanti, elettricisti, tecnici elettronici e chiunque altro lavori con i circuiti.

Appendice F: Notizie Ulteriori Circa L'Elettricità: questa appendice contiene un glossario di termini ed un Esercizio che potete provare per impratichirvi con la misura della tensione, della corrente e della resistenza.

Il Vostro Turno – Modificare il circuito di prova del LED.

Nel prossimo Esercizio, programmerete il BASIC Stamp per accendere il LED, quindi per spengerlo, e così via. Il BASIC Stamp lo farà collegando il circuito del LED a due diversi connettori, Vdd e Vss. Avete appena finito di lavorare al circuito dove la resistenza è collegata a Vdd, ed il LED emette luce. Fate i cambiamenti mostrati nella Figura 2-10 per verificare che il LED si spengerà (non emetterà luce) quando il terminale della resistenza è scollegato da Vdd e collegato a Vss.

√ Scollegare l'alimentazione dalla vostra Board of Education o HomeWork Board.
√ Scollegare il terminale della resistenza che è collegato nel connettore Vdd, e collegatelo in un connettore marcato Vss come mostrato nella Figura 2-10.

√ Ricollegate l'alimentazione alla vostra Board of Education o HomeWork Board.

√ Controllate per assicurarvi che il LED verde **non emetta luce**. Non si dovrebbe accendere.

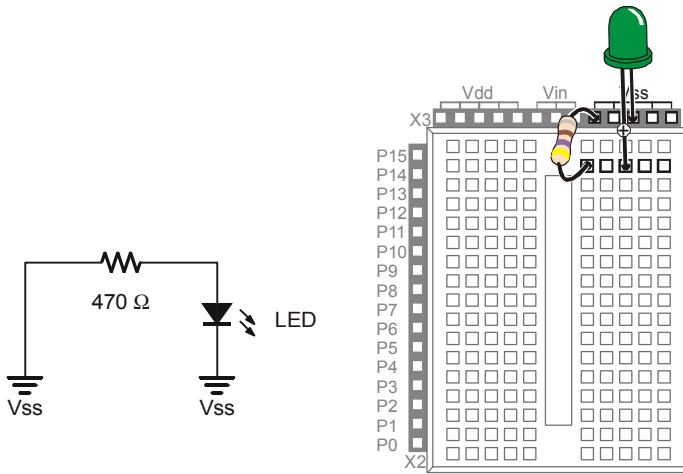



Figura 2-10
Circuito del LED spento

Schema elettrico (sinistra) e schema pratico (destra).

ESERCIZIO #2: CONTROLLO DI ACCENSIONE/SPEGNIMENTO CON IL BASIC STAMP

Nell'Esercizio #1, sono stati costruiti e collaudati due diversi circuiti. Un circuito faceva accendere il LED mentre l'altro No. La Figura 2-11 mostra come il BASIC Stamp possa fare la stessa cosa se collegate un LED ad uno dei suoi piedini di I/O. In quest'Esercizio, collegherete il circuito LED al BASIC Stamp e lo programmerete per accenderlo e spengerlo. Sperimenterete inoltre con programmi che lo faranno fare a diverse velocità.

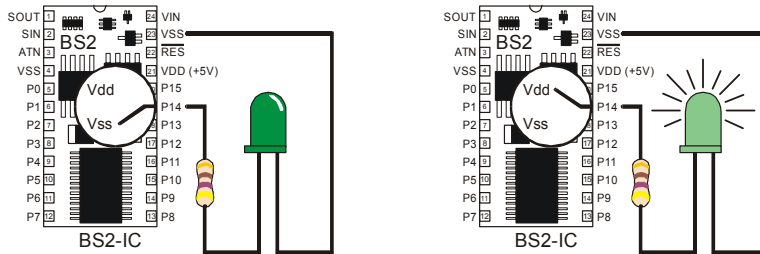



Figura 2-11
Commutazione del BASIC Stamp
Il BASIC Stamp può essere programmato per collegare internamente l'ingresso del circuito LED a Vdd o a Vss.

Ci sono due grandi differenze tra cambiare manualmente la connessione ed avere il BASIC Stamp che lo fa. Primo, il BASIC Stamp non deve togliere l'alimentazione quando cambia il collegamento da Vdd a Vss. Secondo, mentre un uomo può fare questo scambio diverse volte al minuto, il BASIC Stamp lo può fare migliaia di volte al secondo!

Componenti del circuito di prova del LED

Gli stessi dell'Esercizio #1.

Collegare il circuito del LED al BASIC Stamp.

Il circuito del LED mostrato nella Figura 2-12 è collegato quasi nella stessa maniera del circuito dell'esercizio precedente. La differenza è che il terminale della resistenza che era manualmente collegato a Vdd e poi a Vss, è ora inserito in un piedino di I/O del BASIC Stamp.

- √ Scollegare l'alimentazione dalla vostra Board of Education o HomeWork Board.
- √ Modificate il circuito su cui stavate lavorando nell'Esercizio #1 cosicché sia uguale a quello di Figura 2-12.

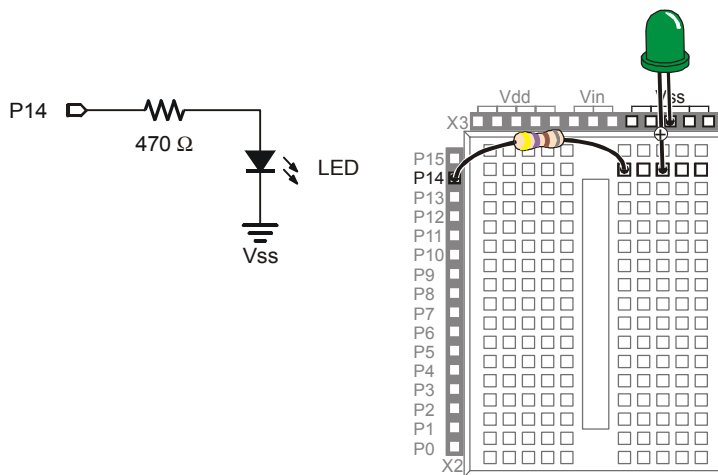



Figura 2-12
Circuito LED con il BASIC Stamp

L'ingresso del circuito LED è ora collegato ad un piedino di I/O del BASIC Stamp invece che a Vdd o Vss.

Le resistenze sono essenziali. Ricordare sempre di usare una resistenza. Senza di essa, nel circuito scorrerà troppa corrente, e questo può danneggiare i componenti del vostro circuito, i componenti del BASIC Stamp, o della Board of Education o della HomeWork Board.

Accendere/Spengere il LED con un programma.

Il programma esempio fa lampeggiare il LED una volta al secondo. Questa cosa introduce diverse tecniche di programmazione tutte in una volta. Dopo averlo lanciato, esperimenterete con diverse parti del programma per meglio capire come funziona.

Programma di Esempio: LedOnOff.bs2

√ Digitate il codice LedOnOff.bs2 nell'Editor del BASIC Stamp.
√ Rialimentate la vostra Board of Education o HomeWork Board.
√ Lanciate il programma.
√ Verificate che il LED lampeggi ad un ritmo di un secondo.
√ Quando avete finito, scollegate l'alimentazione.

```
' Che cosa è un Microcontrollore - LedOnOff.bs2
' Accende e spenge un LED. Lo ripete 1 volta al secondo indefinitamente.

' {$STAMP BS2}
' {$PBASIC 2.5}

DO

  HIGH 14
  PAUSE 500
  LOW 14
  PAUSE 500

LOOP
```

Come funziona LedOnOff.bs2

Il comando **`HIGH 14`** forza il BASIC Stamp a collegare internamente il Pin I/O 14 a Vdd. Il LED si accende.

Il comando **`PAUSE 500`** forza il BASIC Stamp a non far niente per ½ secondo mentre il LED resta acceso. Il numero 500 indica al comando **`PAUSE`** di attendere 500/1000 di secondo. Il numero che segue **`PAUSE`** si chiama argomento. Se cercate **`PAUSE`** nel Manuale del BASIC Stamp, scoprirete che questo numero viene chiamato argomento ***`Durata`***. Il

nome durata di quest'argomento indica che il comando **`PAUSE`** fa una pausa di un dato numero di millisecondi.

Che cosa è un Millisecondo? Un millisecondo è 1/1000 di un secondo. La sua abbreviazione è ms. Ci vogliono 1000 ms per fare un secondo.

Il comando **`LOW 14`** forza il BASIC Stamp a collegare internamente il Pin I/O pin P14 a Vss. Questo spenge il LED. Dal momento che **`LOW 14`** è seguito da un altro comando **`PAUSE 500`**, il LED resta spento per ½ secondo.

La ragione per cui il codice si ripete all'infinito, è perché è inserito all'interno dei comandi PBASIC **`DO`** e **`LOOP`**. La Figura 2-13 mostra come funziona un comando **`DO…LOOP`**. Inserendo il segmento di codice che accende e spenge il LED con le pause tra **`DO`** e **`LOOP`**, diciamo al BASIC Stamp di eseguire questi quattro comandi all'infinito. Il risultato è che il LED lampeggerà all'infinito. Continuerà a lampeggiare fino a che toglierete l'alimentazione, premerete il tasto reset, o fino a che la pila non sarà scarica.

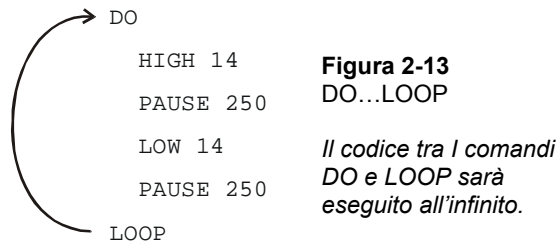



Figura 2-13
DO…LOOP

Il codice tra I comandi DO e LOOP sarà eseguito all'infinito.

Il Vostro Turno – Temporizzazioni e Ripetizioni.

Cambiando l'argomento ***`Durata`*** del comando **`PAUSE`** potrete cambiare la quantità di tempo in cui il LED sta acceso e spento. Per esempio, cambiando a 250 l'argomento ***`Durata`*** di ambedue i comandi **`PAUSE`**, farà lampeggiare il LED due volte al secondo. **`DO…LOOP`** nel vostro programma saranno adesso come di seguito:

```
DO
  HIGH 14
  PAUSE 250
  LOW 14
  PAUSE 250
LOOP
```

√ Modificare l'argomento ***Durata*** del comando **PAUSE** cosicché siano 250, e riavviare il programma.

Se volete far lampeggiare il LED ogni tre secondi, con il LED spento per un tempo doppio del LED acceso, potrete programmare il comando **PAUSE** dopo il comando **HIGH 14** in modo che duri un secondo usando **PAUSE 1000**. Il comando **PAUSE** dopo il comando **LOW 14** dovrà essere **PAUSE 2000**.

```
DO

  HIGH 14
  PAUSE 1000
  LOW 14
  PAUSE 2000

LOOP
```

√ Modificate e rilanciate il programma con il frammento di codice mostrato sopra.

Un esperimento divertente consiste nel vedere quanto potete accorciare le pause pur continuando a vedere lampeggiare il LED. Quando il LED lampeggia velocemente,ma sembra essere sempre acceso, questo fenomeno si chiama persistenza visiva. Di seguito viene indicata la procedura per vedere qual è la vostra soglia di persistenza visiva:

√ Provate a modificare ambedue le ***Durate*** dei comandi **PAUSE** così che siano 100.
√ Rilanciate il programma e controllate per vedere un rapido lampeggio.
√ Riducete ambedue gli argomenti ***Durata*** di 5 e provate ancora.
√ Continuate a ridurre gli argomenti ***Durata*** fino a che il LED appaia sempre acceso senza lampeggi. Dovrebbe accendersi in modo attenuato ma senza lampeggiare.

Un'ultima cosa da provare, è far fare un unico lampo al LED. Quando il programma gira, il LED lampeggia una sola volta. Questo è un modo di verificare la funzionalità di **DO…LOOP**. Potete rimuovere temporaneamente il **DO…LOOP** dal programma mettendo un apostrofo a sinistra delle parole **DO** e **LOOP** come mostrato sotto.

```
' DO

  HIGH 14
  PAUSE 1000
  LOW 14
```

2

```
        PAUSE 2000
    ' LOOP
```

√ Modificate e rilanciate il programma usando il frammento di codice scritto sopra.
√ Spiegate che cosa è successo, perché il LED si è acceso solo una volta?

Commentare una linea di codice: Mettere un apostrofo a sinistra di un comando, lo trasforma in un commento. Questo è uno strumento utile perché non dovrete fisicamente cancellare il codice per vedere cosa accade se lo togliete dal programma. È molto più facile aggiungere e rimuovere un apostrofo che eliminare e ridigitare i comandi.

ESERCIZIO #3: CONTEGGIO E RIPETIZIONE

Nell'Esercizio precedente, il circuito LED lampeggiava di continuo oppure una sola volta. Ma se lo volessimo far lampeggiare dieci volte? I Computer (incluso il BASIC Stamp) sono bravi nel tenere il conto di quante volte qualche cosa è accaduto. I Computer possono anche essere programmati per prendere decisioni basate su una moltitudine di condizioni. In quest'Esercizio, programmerete il BASIC Stamp per far smettere il lampeggio del LED dopo dieci lampeggi.

Componenti necessari

Gli stessi dell'Esercizio precedente

Circuito di Prova

Usare il circuito esempio mostrato nella Figura 2-12 a pagina 51.

Quante Volte?

Ci sono molte maniere di far lampeggiare un LED dieci volte. Il modo più semplice è usare un ciclo **FOR…NEXT**. Il ciclo **FOR…NEXT** è simile al ciclo **DO…LOOP**. Sebbene ambedue I comandi possano essere usati per ripetere i comandi un numero definito di volte, **FOR…NEXT** è più facile da usare.

Il ciclo **FOR…NEXT** dipende da una variabile per tenere traccia di quante volte il LED ha lampeggiato. Una variabile è una parola di vostra scelta che viene usata per ricordare un valore. Il prossimo esempio usa la parola **counter** per 'contare' quante volte il LED ha lampeggiato.

Scegliere le parole per I nomi delle variabili ha alcune regole:

1. Il nome non può essere una parola già usata dal PBASIC. Queste parole sono chiamate parole riservate, ed alcuni esempi che dovreste già conoscere sono: **DEBUG**, **PAUSE**, **HIGH**, **LOW**, **DO**, e **LOOP**.
2. Il nome non può contenere uno spazio.
3. Sebbene il nome possa contenere lettere, numeri, e sottolineati, deve cominciare con un carattere.
4. Il nome deve essere lungo non più di 33 caratteri.

Programma di Esempio: LedOnOffTenTimes.bs2

Il programma LedOnOffTenTimes.bs2 dimostra come usare un ciclo **FOR…NEXT** per far lampeggiare un LED dieci volte.

√ Il vostro circuito di test dell'Esercizio #2 deve essere costruito (o ricostruito) e pronto all'uso.
√ Digitate il codice LedOnOffTenTimes.bs2 nell'Editor del BASIC Stamp.
√ Accendete la vostra Board of Education o HomeWork Board.
√ Attivate il programma.
√ Verificate che il LED lampeggi dieci volte.
√ Attivate il programma una seconda volta, e verificate che il valore di **counter** mostrato nel Terminale di Debug segua accuratamente quante volte il LED ha lampeggiato. Suggerimento: invece di clickare *Run* una seconda volta, potete premere il tasto reset sulla vostra Board of Education o HomeWork Board.

```
' Che cosa è un Microcontrollore- LedOnOffTenTimes.bs2
' Accende e spenge un LED. Ripete 10 volte.

' {$STAMP BS2}
' {$PBASIC 2.5}

counter VAR Byte

FOR counter = 1 TO 10

  DEBUG ? counter

  HIGH 14
  PAUSE 500
  LOW 14
  PAUSE 500
```

2

```
NEXT

DEBUG "All done!"

END
```

Come funziona LedOnOffTenTimes.bs2

Questa dichiarazione PBASIC

```
counter VAR Byte
```

Dice all'Editor del BASIC Stamp che il vostro programma userà la parola **counter** come variabile che può contenere un'informazione grande un byte.

Che cosa è un Byte? Un byte è abbastanza memoria da contenere un numero da 0 a 255. Il BASIC Stamp ha quattro diversi tipi di variabili, e ciascuna può contenere una gamma diversa di numeri:

Tabella 2-2: Tipi di Variabili e Valori che possono

Tipi di Variabile	Gamma dei Valori
Bit	0 a 1
Nib	0 a 15
Byte	0 a 255
Word	0 a 65535

Il punto interrogativo (formattatore) prima di una variabile in un comando **DEBUG** dice al Terminale di Debug di visualizzare il nome della variabile ed il suo valore. Di seguito è mostrato come il comando

```
DEBUG ? counter
```

Visualizza sia il nome che il valore della variabile **counter** nel Terminale di Debug.

Il ciclo **FOR…NEXT** e tutti i comandi al suo interno sono mostrati sotto. La dichiarazione **FOR counter = 1 to 10** dice al BASIC Stamp che dovrà impostare la variabile **counter** a 1, quindi eseguire tutti i comandi fino a che arriva all'istruzione **NEXT**. Quando il BASIC Stamp arriva all'istruzione **NEXT**, ritorna all'istruzione **FOR.** L'istruzione **FOR** aggiunge uno al valore di **counter**. Quindi controlla per vedere se **counter** è maggiore di dieci. Se no, ripete il procedimento. Se il valore di counter

finalmente raggiunge undici, il programma scarta i comandi tra le istruzioni **`FOR`** e **`NEXT`** e va al comando immediatamente dopo l'istruzione **`NEXT`**.

```
FOR counter = 1 to 10

  DEBUG ? counter, CR

  HIGH 14
  PAUSE 500
  LOW 14
  PAUSE 500

NEXT
```

Il comando che viene dopo l'istruzione **`NEXT`** è:

```
DEBUG "Tutto fatto!"
```

Questo comando viene inserito solamente per mostrare che cosa fa il programma dopo essere passato dieci volte attraverso il ciclo **`FOR…NEXT`**. Va al comando che viene dopo l'istruzione **`NEXT`**.

Il Vostro Turno – Altre maniere di contare

√ Nel programma LedOnOffTenTimes.bs2 sostituite l'istruzione

```
FOR counter = 1 to 10
```

Con l'istruzione

```
FOR counter = 1 to 20
```

E farlo ripartire. Che cosa fa di diverso, e questo era previsto?

√ Provate una seconda modifica all'istruzione **`FOR`**. Questa volta, cambiatela in

```
FOR counter = 20 to 120 STEP 10
```

Quante volte ha lampeggiato il LED? Quali valori sono stati visualizzati nel Terminale di Debug?

ESERCIZIO #4: COSTRUIRE E PROVARE UN SECONDO CIRCUITO LED

Gli indicatori a LED possono essere usati per indicare molte cose agli utilizzatori della macchina. Molti dispositivi abbisognano di due, tre o più LED per indicare all'utente se la macchina è pronta o no, se c'è un malfunzionamento, se ha eseguito un processo e così via.

In quest'Esercizio, ripeterete il circuito LED dell'Esercizio #1 per un secondo circuito LED. Quindi adatterete il programma esempio dell'Esercizio #2 per assicurarvi che il circuito LED sia connesso correttamente al BASIC Stamp. Dopodiché, modificherete il programma esempio dell'Esercizio #2 per far lavorare in tandem i due LED.

Componenti aggiuntivi

In aggiunta ai componenti che avete usato negli esercizi 1 e 2, vi serviranno i seguenti componenti:

(1) LED - giallo
(1) Resistenza – 470 Ω (giallo-viola-marrone)

Costruzione e Prova del Secondo Circuito LED.

Nell'Esercizio #1, avete provato manualmente il primo circuito LED per assicurarvi che funzionasse prima di collegarlo al BASIC Stamp. È importante, prima di collegarlo al BASIC Stamp, collaudare anche il secondo circuito LED.

√ Scollegare l'alimentazione dalla vostra Board of Education o HomeWork Board.
√ Costruite il secondo circuito LED come mostrato in Figura 2-14.
√ Rialimentate la vostra Board of Education o HomeWork Board.
√ Il circuito LED che avete appena aggiunto si accende? Se si, continuare. Se no, nell'Esercizio #1 ci sono alcuni suggerimenti per la ricerca guasti che potete ripetere per questo circuito.

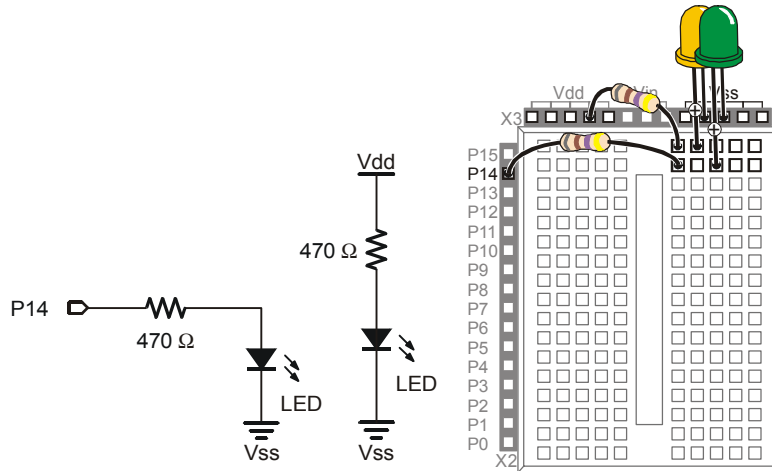



Figura 2-14
Circuito di collaudo manuale per il secondo LED

- √ Spengete la vostra Board of Education o HomeWork Board
- √ Modificate il secondo circuito LED che avete appena collaudato, collegando il terminale della resistenza del LED (ingresso) a P15 come mostrato in Figura 2-15.

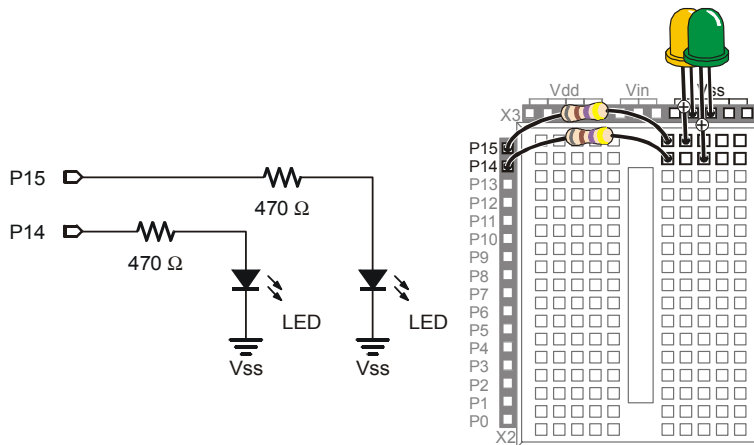



Figura 2-15
Collegare il secondo LED al BASIC Stamp

Schema elettrico (sinistra) e schema pratico (destra).

2

Usare un programma per collaudare il secondo circuito LED.

Nell'Esercizio #2, avete usato un programma esempio ed i comandi **HIGH** e **LOW** per controllare il circuito LED collegato a P14. Questi comandi devono essere modificati per controllare il circuito LED collegato a P15. Invece di usare **HIGH 14** e **LOW 14**, userete **HIGH 15** e **LOW 15**.

Programma di Esempio: TestSecondLed.bs2

√ Digitare il programma TestSecondLed.bs2 nell'Editor del BASIC Stamp.
√ Accendete la vostra Board of Education o HomeWork Board.
√ Avviare TestSecondLED.bs2.
√ Assicuratevi che il circuito LED collegato a P15 lampeggi. Se il LED collegato a P15 lampeggia, andare al prossimo esempio (Controllare ambedue i LED). Se il circuito LED collegato a P15 non lampeggia, controllate il vostro circuito per errori di cablaggio ed il vostro programma per errori di battitura e provate di nuovo.

```
' Che cosa è un Microcontrollore- TestSecondLed.bs2
' Accende e spenge il LED collegato a P15.
' Ripete 1 volta al secondo indefinitamente.

' {$STAMP BS2}
' {$PBASIC 2.5}

DO

  HIGH 15
  PAUSE 500
  LOW 15
  PAUSE 500

LOOP
```

Controllare ambedue i LED

Ebbene si, potete far lampeggiare ambedue i LED insieme. Una maniera in cui potete farlo è di usare due comandi **HIGH** prima del comando **PAUSE**. Un comando **HIGH** imposta P14 alto, ed il successivo comando **HIGH** imposta P15 alto. Dovrete anche inserire due comandi **LOW** per spengere entrambi i LED. In verità i due LED non lampeggeranno esattamente nello stesso momento perché saranno accesi o spenti uno dopo l'altro. D'altronde, c'è una differenza di non più di un millisecondo tra I due cambiamenti, e l'occhio umano non la noterà.

Programma Esempio: FlashBothLeds.bs2

√ Digitate il codice FlashBothLeds.bs2 nell'Editor del BASIC Stamp.
√ Avviate il program.
√ Verificate che ambedue i LED lampeggino nello stesso momento.

```
' Che cosa è un Microcontrollore- FlashBothLeds.bs2
' Accende e spenge i LED collegati a P14 e P15.

' {$STAMP BS2}
' {$PBASIC 2.5}

DO

  HIGH 14
  HIGH 15
  PAUSE 500
  LOW 14
  LOW 15
  PAUSE 500

LOOP
```

Il Vostro Turno – Alternare l'accensione dei LED.

Potrete far accendere alternativamente i LED scambiando i comandi **HIGH** e **LOW** che controllano i pin I/O. Questo significa che mentre un LED è acceso, l'altro è spento.

√ Modificate FlashBothLeds.bs2 cosicché i comandi tra le istruzioni **DO** e **LOOP** siano questi:

```
HIGH 14
LOW 15
PAUSE 500
LOW 14
HIGH 15
PAUSE 500
```

√ Avviare la versione modificata di FlashBothLeds.bs2 e verificate che i LED lampeggino alternativamente.

ESERCIZIO #5: USARE IL VERSO DELLA CORRENTE PER CONTROLLARE UN LED BICOLORE

Il dispositivo mostrato nella Figura 2-16 è un visualizzatore di sicurezza per chiavi elettroniche. Quando viene usata la chiave elettronica con il giusto codice, il LED cambia colore, ed una porta si apre. Questo tipo di LED si chiama LED bicolore. Questo Esercizio risponde a due domande:

1. Come cambia colore un LED?
2. Come funziona con il BASIC Stamp?

Figura 2-16
LED Bicolore in un dispositivo di sicurezza

Quando la porta è chiusa, questo LED bicolore s'illumina di rosso. Quando la porta è aperta con una chiave elettronica con il codice giusto, il LED s'illumina di verde.

Conosciamo il LED BiColore

Il simbolo schematico del LED bicolore ed il disegno sono mostrati nella Figura 2-17.

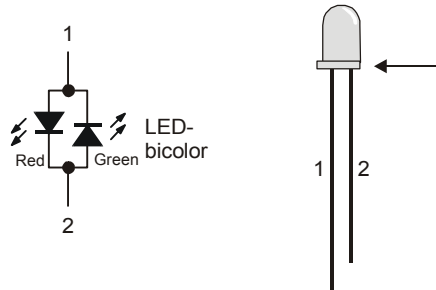



Figura 2-17
LED Bicolore

Simbolo schematico (sinistra) e disegno del componente (destra).

Il LED bicolore in realtà è composto di due LED nello stesso involucro. La Figura 2-18 mostra che potete alimentarlo in un verso e si accenderà il LED rosso. Collegandolo al contrario, si accenderà il LED verde. Come per gli altri LED, se collegate ambedue I terminali a Vss, il LED non si accenderà.

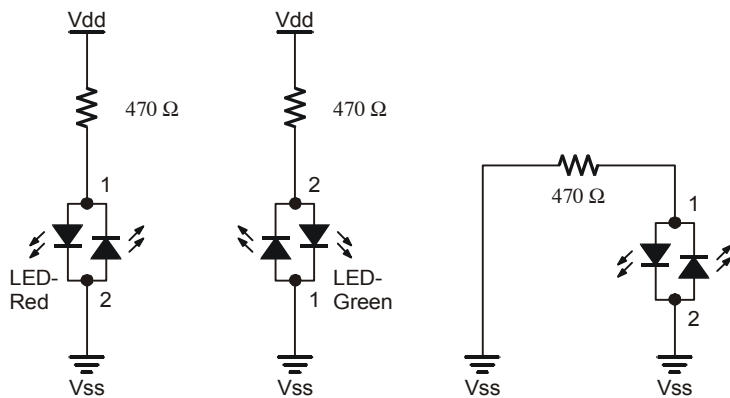



Figura 2-18
LED Bicolore e modalità di alimentazione

Rosso (sinistra), verde (centro) e spento (destra).

Componenti per il circuito del LED Bicolore

(1) LED – bicolore
(1) Resistenza 470 Ω (giallo-viola-marrone)
(1) Ponticello di filo

Costruzione e Collaudo del Circuito LED Bicolore.

La Figura 2-19 illustra la prova manuale per il LED bicolore.

- √ Scollegare l'alimentazione dalla vostra Board of Education o HomeWork Board.
- √ Costruire il circuito mostrato a sinistra della Figura 2-19.
- √ Riaccendete e verificate che il LED bicolore si accenda di rosso.
- √ Spengete di nuovo.
- √ Modificate il circuito che corrisponde al disegno a destra della Figura 2-19.
- √ Riaccendete.
- √ Verificate che il LED bicolore si accenda di verde.
- √ Spengete.

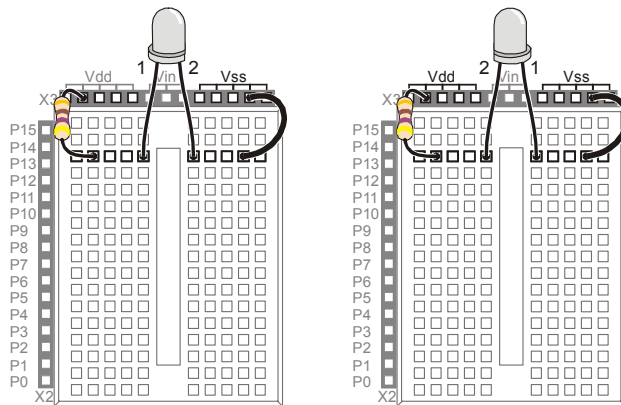



Figura 2-19
Collaudo Manuale del LED bicolore
LED Bicolore rosso (sinistra) e verde (destra).

Controllare un LED bicolore con il BASIC Stamp richiede due pin I/O. Dopo che avrete manualmente verificato che il LED bicolore funzioni, potrete collegarlo al BASIC Stamp come mostrato in Figura 2-20.

- √ Collegate il LED bicolore al BASIC Stamp come mostrato nella Figura 2-20.

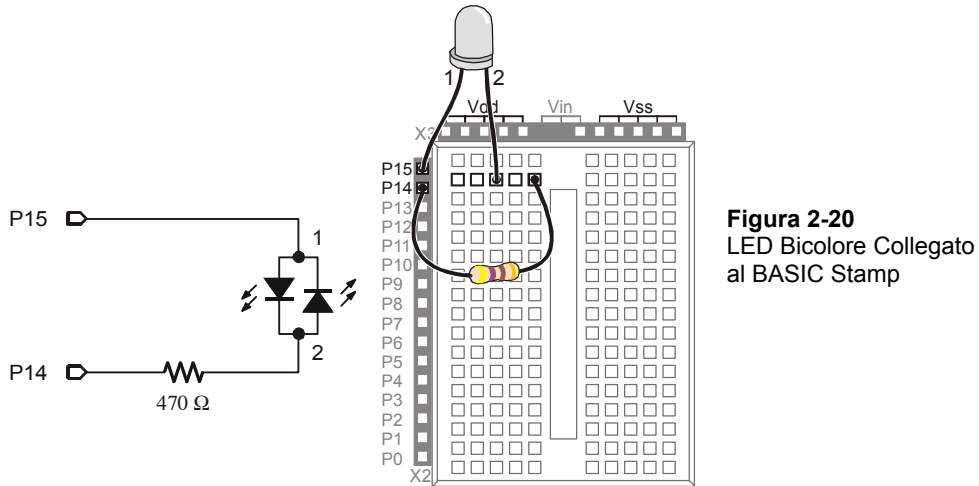



Figura 2-20
LED Bicolore Collegato al BASIC Stamp

Controllo del LED Bicolore tramite il BASIC Stamp.

La Figura 2-21 mostra come potete usare P15 e P14 per controllare il flusso di corrente nel circuito del LED bicolore. Lo schema superiore mostra come scorre la corrente nel LED rosso quando P15 è collegato a Vdd e P14 è collegato a Vss. Questo perché il LED rosso farà scorrere la corrente quando si applica la tensione come mostrato, mentre il LED verde comportandosi come un rubinetto chiuso non permetterà alla corrente di scorrere. Il LED bicolore si accenderà di rosso.

Lo schema inferiore mostra che cosa succede quando P15 è collegato a Vss e P14 è collegato a Vdd. La tensione è ora invertita. Il LED rosso si spenge e non permette alla corrente di scorrere. Nel frattempo il LED verde si accende, la corrente può passare nel circuito nella direzione opposta.

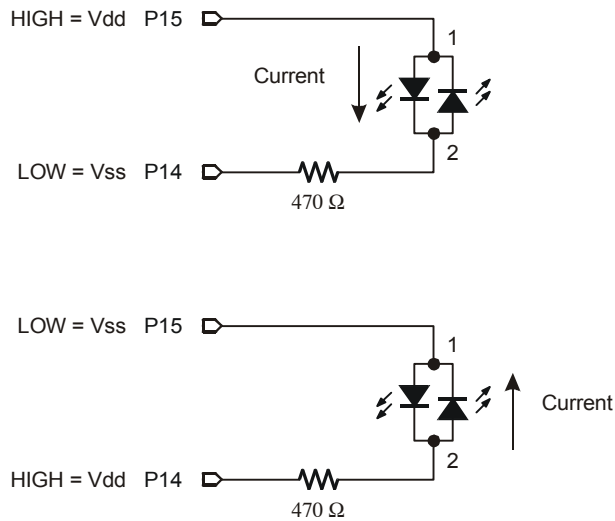



Figura 2-21
Collaudo manuale del LED bicolore

La Corrente attraverso il LED rosso (sopra) ed attraverso il LED verde (sotto).

La Figura 2-21 mostra inoltre il trucco per programmare il BASIC Stamp per fare accendere con due colori diversi il LED bicolore. Lo schema superiore mostra come far accendere il LED bicolore di rosso usando **`HIGH 15`** e **`LOW 14`**. Lo schema inferiore mostra come far accendere il LED bicolore di verde usando **`LOW 15`** e **`HIGH 14`**. Per spengere il LED, inviare un segnale basso a P14 e P15 usando **`LOW 15`** e **`LOW 14`**. In altre parole, usare **`LOW`** su entrambi i pin.

Il LED bicolore si spengerà anche se invierete un segnale alto a P14 e P15. perché? Perché la tensione sarà la stessa su P14 e P15 non importa se imposterete ambedue I pin high o low, in ogni caso non ci sarà differenza di potenziale.

Programma di Esempio: TestBiColorLED.bs2

√ Riaccendete
√ Digitate il codice TestBiColorLed.bs2 nell'Editor del BASIC Stamp.
√ Avviate il programma.
√ Verificate che il LED si alterni attraverso gli stati rosso, verde, spento.

```
' Che cosa è un Microcontrollore- TestBiColorLed.bs2
' Accende il LED bicolore rosso, quindi verde, quindi spento in sequenza.
```

```
' {$STAMP BS2}
' {$PBASIC 2.5}

DO

  HIGH 15           ' Rosso
  LOW 14
  PAUSE 500

  LOW 15            ' Verde
  HIGH 14
  PAUSE 500

  LOW 15            ' Off
  LOW 14
  PAUSE 500

LOOP
```

Il Vostro Turno – Visualizzazione Luminosa

Nell'Esercizio #3, una variabile chiamata **`counter`** era usata per controllare quante volte il LED lampeggiava. Che cosa succederebbe se usaste il valore **`counter`** per controllare l'argomento ***`Durata`*** del comando **`PAUSE`**?

√ Salvare TestBiColorLed.bs2 con un nuovo nome, come per esempio TestBiColorLedYourTurn.bs2.

√ Aggiungete una dichiarazione di variabile counter prima dell'istruzione **`DO`**:

```
counter VAR BYTE
```

√ Quindi il ciclo **`FOR…NEXT`** mostrato sotto all'interno del **`DO…LOOP`**.

```
FOR counter = 1 to 50

  HIGH 15
  LOW 14
  PAUSE counter

  LOW 15
  HIGH 14
  PAUSE counter

NEXT
```

Quando avete finito, il vostro codice deve essere:

```
counter VAR BYTE

DO

  FOR counter = 1 to 50

    HIGH 15
    LOW 14
    PAUSE counter

    LOW 15
    HIGH 14
    PAUSE counter

  NEXT

LOOP
```

All'inizio di ciascun passaggio nel ciclo **FOR…NEXT**, il valore **PAUSE** (argomento ***Durata***) è solamente un millisecondo. Ad ogni passaggio nel ciclo **FOR…NEXT**, la pausa si allunga di un millisecondo alla volta fino a raggiungere i 50 millisecondi. Il ciclo **DO…LOOP** fa eseguire il ciclo **FOR…NEXT** all'infinito.

√ Avviate il programma modificato ed osservate l'effetto.

SOMMARIO

Il BASIC Stamp può essere programmato per accendere e spengere un circuito con un diodo emettitore di luce (LED) (indicatore luminoso). Gli indicatori a LED sono utili in una varietà di posti compresi molti monitor di computer, disk driver ed altri dispositivi. Il LED è stato presentato insieme alla tecnica per identificare i terminali anodo e catodo. Un circuito LED deve avere una resistenza per limitare la corrente che lo attraversa. Le Resistenze sono state presentate insieme con uno dei più comuni sistemi di codifica che potete usare per conoscere il loro valore.

Il BASIC Stamp accende e spenge un circuito LED collegando internamente un pin I/O a Vdd oppure a Vss. Il comando **`HIGH`** può essere usato per far collegare al BASIC Stamp internamente uno dei suoi pin I/O a Vdd, ed il comando **`LOW`** può essere usato per collegare internamente un pin I/O a Vss. Il comando **`PAUSE`** è usato per forzare il BASIC Stamp a non eseguire comandi per un periodo prefissato di tempo. Questo è stato usato per far stare un LED acceso o spento per un certo tempo. Questo periodo di tempo può essere determinato dal numero usato nell'argomento ***`Durata`*** nel comando **`PAUSE`**.

Il comando **`DO…LOOP`** può essere usato per creare un ciclo infinito. I comandi tra le istruzioni **`DO`** e **`LOOP`** saranno eseguiti all'infinito. Sebbene questo venga chiamato ciclo infinito, il programma può sempre essere riavviato spengendo e riaccendendo o premendo e rilasciando il tasto reset. Si può anche scaricare un nuovo programma nel BASIC Stamp, e questo cancellerà il programma con il ciclo infinito.

Usando il LED bicolore sono stati introdotti il verso della corrente e la polarità della tensione. Se la tensione è applicata ai capi del circuito LED, la corrente vi fluirà in una direzione, ed il LED si accenderà del colore corrispondente, se la polarità verrà invertita, la corrente fluirà nella direzione opposta ed il LED si accenderà dell'altro colore.

Domande

1. Quale è il nome di questa lettera greca: Ω?

2. A quale misura si riferisce il simbolo Ω?

3. Come si riconoscono in un LED l'anodo ed il catodo?

4. Quale resistenza permetterà il passaggio di una maggiore corrente attraverso il circuito, una resistenza di 470 Ω o una di 1000 Ω?
5. Quali colori cerchereste sulle fasce di una resistenza per un valore di 2000 Ω?
6. Se la resistenza ha i colori marrone-nero-verde, qual è il suo valore in Ω?
7. Come si collegano due fili usando una piastra per prototipi? Potete usare una piastra per prototipi per collegare insieme quattro fili?
8. Quale componente limita la corrente?
9. Quando la corrente sta scorrendo in un circuito LED, gli elettroni fluiscono fuori da Vdd o da Vss?
10. Che cosa dovete sempre fare prima di modificare un circuito che avete costruito sulla piastra per prototipi?
11. Quanto dura un comando **`PAUSE 500`**?
12. Quanto dovrebbe durare un comando **`PAUSE 10000`**?
13. Come fareste a non fanfare nulla al BASIC Stamp per un intero minuto?
14. Se inserite tre comandi in un ciclo **`DO…LOOP`**, quante volte saranno eseguiti questi comandi?
15. Quale comando potete usare per controllare il numero di volte che un gruppo di comandi viene eseguito?
16. Quali sono i tipi diversi di variabile?
17. Può un byte avere il valore 500?
18. Qual è la variabile più piccola che potete usare per contare fino a 10?
19. Che cosa fa il punto interrogativo in un comando **`DEBUG`**?

20. Che cosa fa il comando **`HIGH 7`**?

21. Il comando **`LOW 30`** è ragionevole? Se si, perché? Se no, che cosa c'è di sbagliato?

22. Se un LED bicolore è collegato a P5 e P9, quali comandi vi serviranno per accenderlo? Quali comandi usereste per fargli cambiare colore? Quale comandi usereste per spengerlo?

Esercizi

1. Disegnate lo schema di un circuito LED simile a quello su cui avete lavorato nell'Esercizio #2, ma collegate il circuito a P13 invece di P14.

2. Spiegate come modifichereste LedOnOff.bs2 a Pagina 52 in modo ché faccia lampeggiare il vostro circuito LED nell'Esercizio 1, quattro volte al secondo.

3. Spiegate come modificare LedOnOffTenTimes.bs2 cosicché faccia lampeggiare il circuito LED 5000 volte prima di fermarsi. Suggerimento: dovrete modificare solamente due linee di codice.

Progetti

1. Fate un contatore inverso da 10 secondi usando un LED giallo ed un LED bicolore. Fate in modo che il LED bicolore cominci dal rosso per tre secondi. Dopo 3 secondi il colore diventi verde. Quando il LED bicolore diventa verde, far lampeggiare il LED giallo una volta al secondo per dieci secondi. Quando il LED giallo ha smesso di lampeggiare, il LED bicolore si deve riaccendere di rosso e rimanere acceso.

2. Costruite il vostro indicatore a barre usando quattro singoli LED.

Usare resistenze da 1000 Ω per questo progetto. Il codice colori per una resistenza da 1000 Ω è marrone-nero-rosso. Usando resistenze da 1000 Ω non fate un indicatore a barre con più di 8 LED. Se volete fare un indicatore a barre con più di 8 LED, usate resistenze da 2000 Ω. Il codice colore per le resistenze da 2000 Ω è rosso-nero-rosso.

Scrivete un programma che comincia tenendo spenti tutti i LED, quindi I LED cominciano ad accendersi uno per volta con un ritardo di un secondo tra ciascun LED. I LED dovrebbero accendersi da destra a sinistra o dall'alto in basso a secondo di come li avete disposti. Salvate il programma con un nome diverso e fate accendere i LED dell'indicatore a barra 20 volte più veloci.

3. Costruite un temporizzatore per semafori. Assumete che i conducenti che provengono sia da nord che da sud possono vedere le luci dei LED Verdi, giallo e rosso collegati a P13, P14, e P15. Fate un altro banco di LED collegati a P3, P4, e P5 che controllano il traffico est-ovest.

   Scrivete un programma che dia al traffico nord-sud 30 secondi di verde, quindi 10 secondi di giallo, seguito da 40 secondi di rosso. Quando il traffico nord-sud è rosso, il traffico est-ovest dovrebbe essere verde per 30 secondi, quindi giallo per 10 secondi poi rosso per 40 secondi.

   Ricordatevi che se questo fosse un semaforo reale, sareste responsabili dei problemi di traffico se ambedue le luci rosse stessero accese contemporaneamente. Se ambedue le luci verdi fossero accese contemporaneamente, potreste causare degli incidenti!

Ulteriori Ricerche

Le risorse elencate sotto, sono disponibili gratuitamente dal sito web della Parallax e sono altresì incluse ne CD della Parallax.

"*Manuale del BASIC Stamp*", Users Manual, Version 2.0c, Parallax Inc., 2000

Il *Manuale del BASIC Stamp* contiene altri esempi che potete provare ed informazioni che spiegano ulteriormente i seguenti comandi: **HIGH**, **LOW**, **PAUSE**, il formattatore **?** del comando **DEBUG** , e **FOR…NEXT**.

"*Basic Analog and Digital*", Student Guide, Version 2.0, Parallax Inc., 2003

Basic Analog and Digital usa i LED per descrivere il conteggio binario, descrive i livelli analogici, e presenta nuove tecniche per regolare la luminosità dei LED.

“*Editor del BASIC Stamp File di aiuto*”, PBASIC 2.5 Version 2.0 Parallax Inc., 2003

Il File di aiuto del PBASIC 2.5 contiene informazioni sul commando **DO…LOOP**, è un nuovo comando del PBASIC 2.5 e non è incluso nel Manuale del BASIC Stamp. Potete trovare questa informazione clickando l’icona libro nella barra delle funzioni dell’Editor del BASIC Stamp, quindi selezionando PBASIC Reference dal menu nella barra laterale sinistra. Questo aprirà la lista alfabetica dei comandi PBASIC nella finestra principale. Informazioni dettagliate possono essere trovate clckando su ciascun comando.

Capitolo #3: Ingressi Digitali - Tasti

3

PRESENTI SU CALCOLATRICI, VIDEOGIOCHI E ALTRE APPLICAZIONI

Quanti dispositivi con i tasti usate giornalmente? Alcuni esempi che possono comparire nel vostro elenco sono: computer, mouse, calcolatrici, forni a microonde, telecomandi, videogiochi, e videoregistratori. In ciascun dispositivo, c'è un microcontrollore che controlla la tastiera ed attende che nel circuito ci sia un cambiamento. Quando il circuito cambia, il microcontrollore rileva il cambiamento ed esegue un'azione. Alla fine di questo Capitolo, sarete esperti nella progettazione dei circuiti con tastiera e programmazione del BASIC Stamp per controllarli ed agire ad un cambiamento di stato.

PARAGONE TRA RICEZIONE E INVIO DEI SEGNALI ALTI E BASSI

Nel Capitolo #2, avete programmato il BASIC Stamp per inviare segnali alti e bassi, ed avete usato i circuiti LED per visualizzare questi segnali. Inviare segnali alti e bassi significa che avete usato i pin I/O del BASIC Stamp come uscite. In questo Capitolo, userete i pin I/O del BASIC Stamp come ingressi. Come ingresso, un pin I/O 'resta in ascolto' di segnali alti o bassi invece di inviarli. Invierete questi segnali al BASIC Stamp usando un circuito con tasti, e programmerete il BASIC Stamp per riconoscere se il tasto è premuto o No.

Altri termini che significano invio, ON/OFF e ricezione: Inviare segnali ON/OFF viene descritto in diversi modi. Potreste vedere l'invio indicato come trasmettere, controllare, o interrompere. Invece di ON/OFF, potreste vederlo chiamare binario, TTL, CMOS, o Booleani. Altri termini per ricezione sono; sentire, rilevare, captare.

ESERCIZIO #1: CONTROLLARE UN TASTO CON UN CIRCUITO LED

Se potete usare un tasto per inviare un segnale alto o basso al BASIC Stamp, potreste anche controllare un LED con un tasto? La risposta è si, ed in quest'Esercizio il LED verrà usato per controllare un tasto.

Conosciamo il tasto

La Figura 3-1 mostra il simbolo schematico ed il disegno pratico di un tasto normalmente aperto. Ad ogni estremità del tasto sono collegati due terminali. Questo significa che collegare un filo al terminale 1 del tasto è la stessa cosa che collegarlo al terminale 4. la stessa regola vale per i terminali 1 e 3. il motivo per cui il tasto non ha solamente due

terminali, è perché al tasto serve stabilità. Se il tasto avesse solamente due terminali, questi terminali si piegherebbero e magari si romperebbero per la pressione che riceve il tasto quando viene ripetutamente premuto.

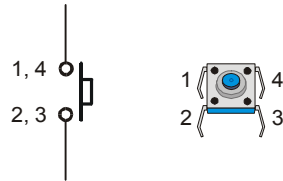



Figura 3-1
Tasto Normalmente Aperto

Simbolo Schematico (sinistra) e disegno del componente (destra)

Il lato sinistro della Figura 3-2 mostra come appare un tasto normalmente aperto quando non è premuto. Quando il tasto non è premuto, c'è uno spazio tra i terminali 1,4 e 2,3. questo spazio fa sì che la corrente non possa fluire dai terminali 1,4 ai terminali 2,3. questo viene chiamato circuito aperto. Il nome "normalmente aperto" significa che lo stato normale del tasto (non premuto) costituisce un circuito aperto. Quando il tasto viene premuto, lo spazio fra i terminali 1,4 e 2,3 viene collegato da una lamina conduttiva. Questo viene chiamato chiusura, e la corrente può fluire attraverso il tasto.

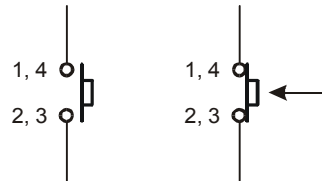



Figura 3-2
Tasto Normalmente Aperto

Non premuto (sinistra) e premuto (destra)

Elenco componenti per il circuito del tasto

(1) LED – un colore qualsiasi
(1) Resistenza - 470 Ω (giallo-viola-marrone)
(1) Tasto - normalmente aperto
(1) Ponticello

Costruire il circuito di prova del Tasto

La Figura 3-3 mostra un circuito che potete costruire per collaudare manualmente il tasto.

Spengete sempre la vostra Board of Education o BASIC Stamp HomeWork Board prima di fare qualsiasi cambiamento al vostro circuito di prova. Da ora in poi, le istruzioni non diranno più "Spengete sempre……." Dopo ogni modifica del circuito. Sta a voi ricordarvi di farlo.

Accendete sempre la vostra Board of Education o BASIC Stamp HomeWork Board prima di scaricare un programma nel BASIC Stamp.

3

√ Costruite il circuito mostrato in Figura 3-3.

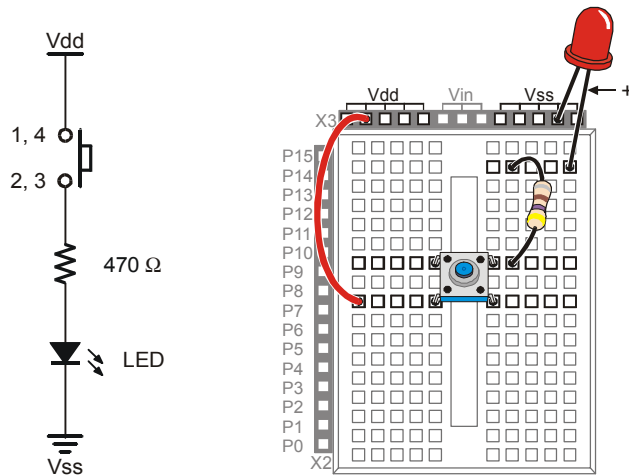



Figura 3-3
Circuito di prova del Tasto

Prova del Tasto

Quando il tasto non è premuto, il LED sarà spento. Se il cablaggio è corretto, quando il tasto viene premuto, il LED si dovrebbe accendere (emettere luce).

Segnali di avvertimento: Se la luce del LED Pwr sulla Board of Education tremola, si attenua, o si spenge completamente quando la rialimentate significa che c'è un corto circuito. Se questo succede, spengete la scheda immediatamente, trovate e correggete l'errore nel vostro circuito.

Il LED Power sulla HomeWork Board è diverso. Esso lampeggia solamente quando un programma sta girando. Se un programma finisce (usando il comando **END**), anche il LED Power si spengerà.

√ Verificate che il LED nel vostro circuito di prova sia spento.

√ Premete e mantenete premuto il tasto, e verificate che il LED emetta luce mentre state tenendo premuto il tasto.

Come funziona il circuito del Tasto.

Il lato sinistro della Figura 3-4 mostra che cosa avviene quando il tasto non è premuto. Il circuito del LED non è collegato a Vdd. È un circuito aperto che non può condurre corrente. Premendo il tasto, chiudete la connessione tra i terminali con una lamina conduttiva creando una via attraverso la quale gli elettroni possono fluire nel circuito.

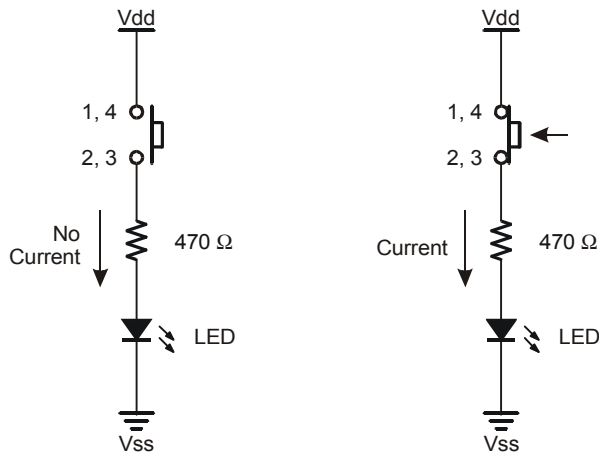



Figura 3-4
Tasto Non Premuto, e Premuto

Circuito del Tasto aperto (sinistra) e chiuso (destra).

Il Vostro Turno – Un Corto Circuito

La Figura 3-5 mostra un circuito diverso che causerà il diverso comportamento del LED. Quando il tasto non è premuto, il LED è acceso. Quando si preme il tasto, il LED si spenge. La ragione per cui il LED si spenge quando il tasto è premuto è perché la corrente segue sempre il percorso che offre la minore resistenza. Quando il tasto viene premuto, i terminali 1,4 e 2,3, praticamente non hanno resistenza tra di loro, così tutta la corrente passa attraverso il tasto (corto circuito) invece che nel LED.

√ Costruite il circuito mostrato in Figura 3-5.
√ Ripetete la prova che avete effettuato nel primo circuito di tasto con questo nuovo circuito.

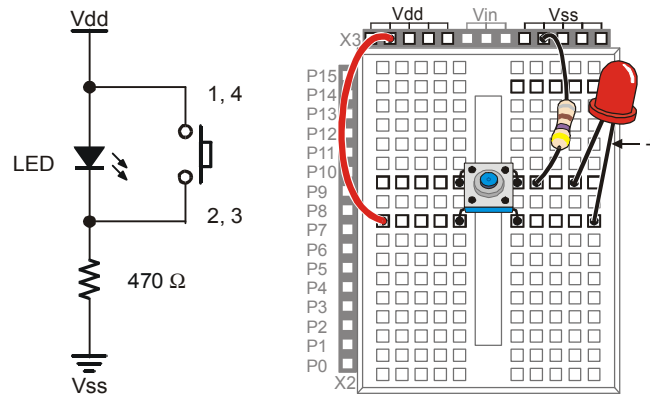



Figura 3-5
LED che viene cortocircuitato dal Tasto

Potete veramente far questo con un LED? Fino ad ora, il catodo del LED è sempre stato collegato a Vss. Ora il LED è collegato in maniera diversa nel circuito, con il suo anodo collegato a Vdd. La gente spesso domanda se questo è contrario alle regole, e la risposta è No. La tensione fornita tra Vdd e Vss è 5 volt. Il diodo userà sempre 1,6 volt, e la resistenza userà sempre 3,4 volt, senza importanza per la loro posizione.

ESERCIZIO #2: RILEVARE UN TASTO CON IL BASIC STAMP

In quest'Esercizio, collegherete un tasto al BASIC Stamp e visualizzerete se il tasto è premuto o No. Lo farete scrivendo un programma PBASIC che controlla lo stato del tasto e lo visualizza nel Terminale di Debug.

Componenti del circuito del Tasto

(1) Tasto - normalmente aperto
(1) Resistenza – 220 Ω (rosso-rosso-marrone)
(1) Resistenza – 10 kΩ (marrone-nero-arancio)
(2) Ponticelli

Costruire un circuito del Tasto per il BASIC Stamp

La Figura 3-6 mostra un circuito di tasto che è collegato al pin I/O P3 del BASIC Stamp.

√ Costruite il circuito mostrato in Figura 3-6.

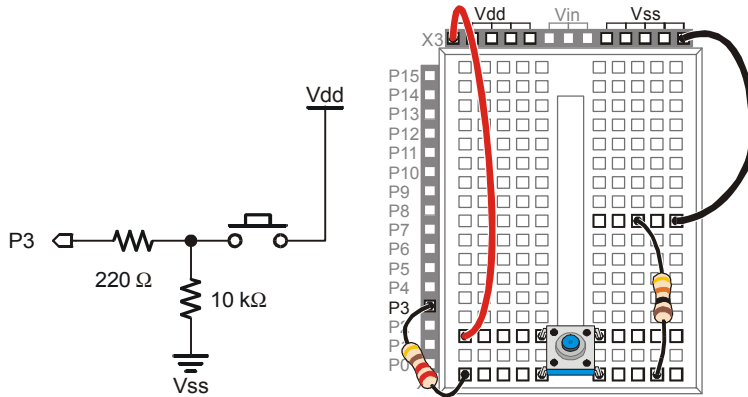



Figura 3-6
Circuito del Tasto Collegato al Pin I/O P3

Sullo schema elettrico, la resistenza da 220 Ω è sul lato sinistro e collega il tasto a P3 mentre la resistenza da 10 kΩ è sulla destra, e collega il circuito del tasto a Vss.

La Figura 3-7 mostra che cosa vede il BASIC Stamp quando il tasto è premuto, e quando non è premuto. Quando il tasto viene premuto, il BASIC Stamp sente che Vdd è collegato a P3. all'interno del BASIC Stamp, questo lo obbliga a mettere 1 in una zona della sua memoria che contiene informazioni circa i suoi pin di I/O. Quando il tasto non è premuto, il BASIC Stamp non può sentire Vdd, ma sente Vss attraverso le resistenze da 10 kΩ e 220 Ω. Questo gli fa memorizzare 0 nella stessa locazione di memoria che conteneva 1 quando il tasto era premuto.

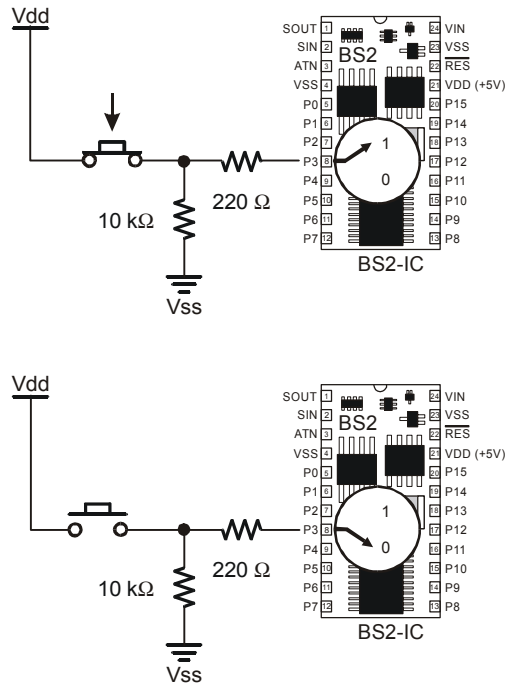



Figura 3-7
Il BASIC Stamp mentre "sente" un Tasto

Quando il tasto viene premuto, il BASIC Stamp sente un 1 (sopra). Quando il tasto non è premuto, il BASIC Stamp sente uno 0 (sotto).

Notazione Binaria e Circuiti: Il sistema numerico a base 2 usa solamente le due cifre 1 e 0 per formare i numeri, e questi valori binari, possono essere trasmessi da un dispositivo all'altro. Il BASIC Stamp interpreta Vdd (5 V) come 1binario e Vss (0 V) come 0 binario. Similmente, quando il BASIC Stamp imposta un pin I/O a Vdd usando **HIGH**, invia un 1 binario. Quando imposta un pin I/O a Vss usando **LOW**, invia uno 0 binario. Questo è un modo molto comune di comunicare numeri binari usato da molti integrati per computer e da altri dispositivi.

Programmare il BASIC Stamp per controllare il Tasto

Il BASIC Stamp memorizza gli 1 e 0 che sente al pin I/O P3 in una locazione di memoria chiamata **IN3**. Qui c'è un programma esempio che mostra come funziona tutto questo:

Programma di Esempio: ReadTastoState.bs2

Questo programma fa controllare al BASIC Stamp il tasto ogni ¼ di secondo ed invia il valore di **IN3** al Terminale di Debug. La Figura 3-8 mostra il Terminale di Debug mentre il programma sta girando. Quando il tasto viene premuto, il Terminale di Debug

visualizza il numero 1, e quando il tasto non è premuto, il Terminale di Debug visualizza il numero 0.

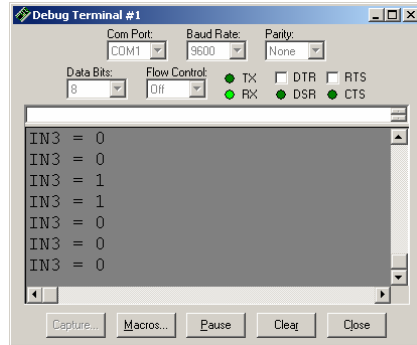



Figura 3-8
Il Terminale di Debug che Visualizza gli stati del Tasto

Il Terminale di Debug visualizza 1 quando il tasto viene premuto e 0 quando non è premuto.

- √ Digitate il programma ReadTastoState.bs2 nell'Editor del BASIC Stamp.
- √ Avviare il programma.
- √ Verificate che il Terminale di Debug visualizzi il valore 0 quando il tasto non è premuto.
- √ Verificate che il Terminale di Debug visualizzi il valore 1 quando il tasto è premuto e tenuto premuto.

```
' Che cosa è un Microcontrollore- ReadTastoState.bs2
' Controlla ed invia lo stato del tasto al Terminale di Debug ogni 1/4 di
' secondo.

' {$STAMP BS2}
' {$PBASIC 2.5}

DO

  DEBUG ? IN3
  PAUSE 250

LOOP
```

Come funziona ReadTastoState.bs2

Il comando **`DO…LOOP`** nel programma si ripete ogni ¼ di secondo a causa del comando **`PAUSE 250`**. A ciascun passaggio attraverso il ciclo **`DO…LOOP`**, il comando **`DEBUG ? IN3`** invia il valore di **`IN3`** al Terminale di Debug. Il valore di **`IN3`** è lo stato in cui si trova il pin I/O P3 all'istante in cui viene eseguito il commando **`DEBUG`**.

Il Vostro Turno – Un Tasto con una Resistenza di Pull-up

Il circuito con cui avete appena finito di lavorare ha una resistenza collegata a Vss. Questa Resistenza viene chiamata Resistenza di pull-down perché quando il tasto non è premuto, mantiene P3 al potenziale di Vss (0 volt). La Figura 3-9 mostra un circuito di tasto che usa una Resistenza di pull-up . Questa Resistenza mantiene la tensione a Vdd (5 volt) quando il tasto non è premuto. La regola è ora invertita, quando il tasto non è premuto, **`IN3`** memorizza il numero 1, e quando il tasto è premuto, **`IN3`** memorizza il numero 0.

La Resistenza da **220 Ω** viene usata nei circuiti esempio del tasto per proteggere il pin I/O del BASIC Stamp. Sebbene questa sia una buona pratica nei prototipi, in molti prodotti, questa Resistenza viene sostituita da un filo (dal momento che il filo costa meno delle resistenze).

√ Modificate il vostro circuito come mostrato in Figura 3-9.
√ Riavviate ReadTastoState.bs2.
√ Usando il Terminale di Debug, verificate che **`IN3`** è 1 quando il tasto non è premuto e 0 quando il tasto è premuto.

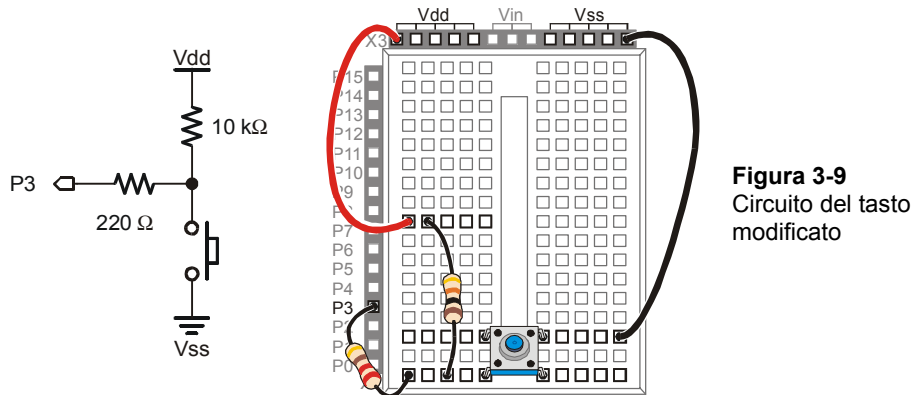



Figura 3-9
Circuito del tasto modificato

Differenze tra Attivo-basso e Attivo-alto: Il tasto in Figura 3-9 viene chiamato attivo-basso perché quando il tasto è attivo (premuto), invia al BASIC Stamp un segnale basso (Vss). Il circuito del tasto di Figura 3-6 nell'Esercizio principale è attivo-alto perché invia un segnale alto (Vdd) quando il tasto è premuto.

ESERCIZIO #3: CONTROLLARE UN CIRCUITO LED CON UN TASTO

La Figura 3-10 mostra una vista ingrandita di un tasto ed un led usati per regolare le impostazioni su un monitor per computer. Questo è appena uno tra tanti dispositivi che hanno un tasto che potete premere per regolare il dispositivo ed un led per mostrare lo stato del dispositivo.

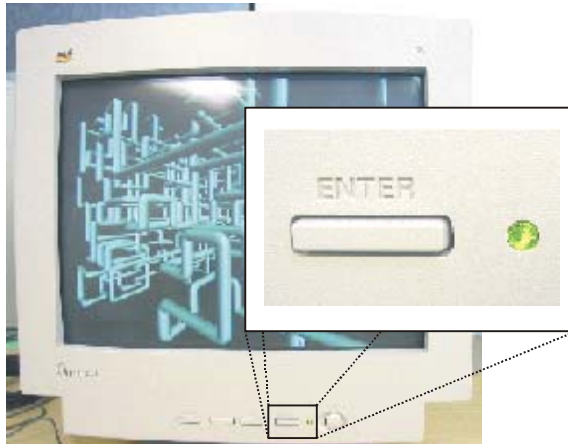



Figura 3-10
Tasto e LED su di un monitor per computer

Il BASIC Stamp può essere programmato per prendere decisioni basate su ciò che sente. Per esempio, può essere programmato per far lampeggiare un LED dieci volte al secondo quando viene premuto un tasto.

Componenti per il circuiti del Tasto e del LED

(1) Tasto – normalmente aperto
(1) Resistenza - 10 kΩ (marrone-nero-arancio)
(1) LED – qualsiasi colore
(1) Resistenza – 220 Ω (rosso-rosso-marrone)
(1) Resistenza – 470 Ω (giallo-viola-marrone)
(2) Ponticelli

3

Costruzione dei circuiti del Tasto e del LED

La Figura 3-11 mostra il circuito del tasto usato nell'Esercizio che avete appena finito insieme con il circuito del LED usato nell'Esercizio #2 Capitolo #2,.

√ Costruite il circuito mostrato in Figura 3-11.

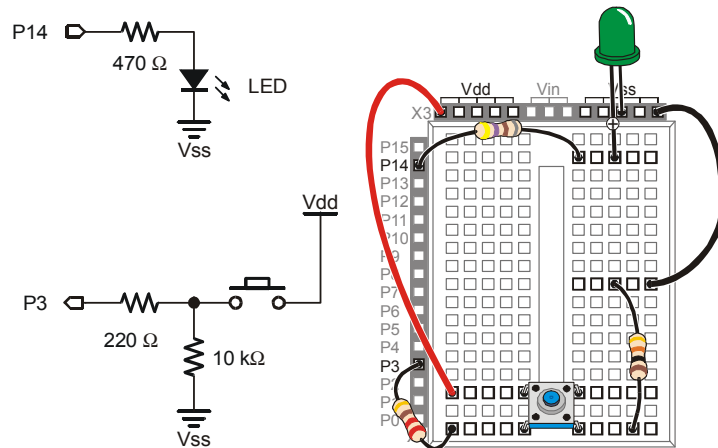



Figura 3-11
Circuito Tasto e LED

Programmare il controllo del Tasto

Il BASIC Stamp può essere programmato per prendere decisioni usando l'istruzione **IF…THEN…ELSE**. Il programma esempio che vi accingete a far girare, farà accendere e spengere il LED quando il tasto viene premuto, usando un'istruzione **IF…THEN…ELSE**. Ad ogni passaggio attraverso il ciclo **DO…LOOP**, l'istruzione **IF…THEN…ELSE** controlla lo stato del tasto e decide se far lampeggiare o no il LED.

Programma di Esempio: TastoControlledLed.bs2

√ Digitate TastoControlledLed.bs2 nell'Editor del BASIC Stamp ed avviatelo.
√ Verificate che il LED lampeggi mentre il tasto è premuto e tenuto premuto.
√ Verificate che il LED non lampeggi quando il tasto non viene premuto.

```
' Che cosa è un Microcontrollore- TastoControlledLed.bs2
' Controlla lo stato del tasto 10 volte al secondo e fa lampeggiare il LED
' quando premuto.
```

```
' {$STAMP BS2}
' {$PBASIC 2.5}

DO

  DEBUG ? IN3

  IF (IN3 = 1) THEN
    HIGH 14
    PAUSE 50
    LOW 14
    PAUSE 50

  ELSE
    PAUSE 100

  ENDIF

LOOP
```

Come funziona TastoControlledLed.bs2

Questo programma è una versione modificata di ReadTastoState.bs2 dell'Esercizio precedente. Il ciclo **DO…LOOP** ed il comando **DEBUG ? IN3** sono gli stessi. Il comando **PAUSE 250** è stato rimpiazzato con un'istruzione **IF…THEN…ELSE**. Quando la condizione dopo l'**IF** è vera **(IN3 = 1)**, sono eseguiti i comandi che vengono dopo l'istruzione **THEN**. Saranno eseguiti fino a che viene raggiunta l'istruzione **ELSE**, a quel punto il programma salta all'istruzione **ENDIF** e continua. Quando la condizione dopo l'istruzione **IF** non è vera **(IN3 = 0)**, sono eseguiti i comandi dopo l'istruzione **ELSE** fino a che viene raggiunta l'istruzione **ENDIF**.

Potete fare una lista dettagliata di che cosa un programma dovrebbe fare, sia per aiutarvi a pianificare il programma sia per descrivere cosa fa. Questo genere di lista viene chiamato pseudo codice, e l'esempio sotto, usa lo pseudo codice per descrivere come funziona il programma.

- *Esegui i comandi da qui all'istruzione Loop all'infinito*
    - *Visualizza il valore di IN3 sul Terminale di Debug*
    - *Se il valore di IN3 è 1, Then*
        - *Accendi il LED*
        - *Attendi 1/20 di secondo*
        - *Spengi il LED*
        - *Attendi 1/20 di secondo*

- *Else, (se il valore di IN3 è 0)*
        - *Non fare nulla, ma attendi per lo stesso tempo che avresti impiegato per far lampeggiare brevemente il LED (1/10 di secondo).*
- *Loop*

Il Vostro Turno – più Veloce/più Lento

√ Salvate il programma esempio con un nome diverso.
√ Modificate il programma così che il LED lampeggi due volte più veloce quando premete e tenete premuto il tasto.
√ Modificate il programma così che il LED lampeggi due volte più lento quando premete e tenete premuto il tasto.

ESERCIZIO #4: DUE TASTI CONTROLLANO DUE CIRCUITI LED

Aggiungete un secondo tasto nel progetto e vedete come lavora. Per rendere le cose un po' più interessanti, aggiungete anche un secondo circuito LED ed usate il secondo tasto per controllarlo.

Componenti del circuito del Tasto e del LED

(2) Tasti – normalmente aperti
(2) Resistenze - 10 kΩ (marrone-nero-arancio)
(2) Resistenze – 470 Ω (giallo-viola-marrone)
(2) Resistenze – 220 Ω (rosso-rosso-marrone)
(2) LED – qualsiasi colore

Aggiungere un Tasto ed un circuito LED

La Figura 3-12 mostra il circuito per un secondo LED e tasto aggiunti al circuito che avete provato nell'Esercizio precedente.

√ Costruite il circuito in Figura 3-12. Se avete bisogno di aiuto nella costruzione del circuito mostrato nello schema, usate come guida lo schema di cablaggio della Figura 3-13.
√ Modificate ReadTastoState.bs2 così che legga lo stato di **`IN4`** invece che di **`IN3`**, ed usatelo per controllare il circuito del secondo tasto.

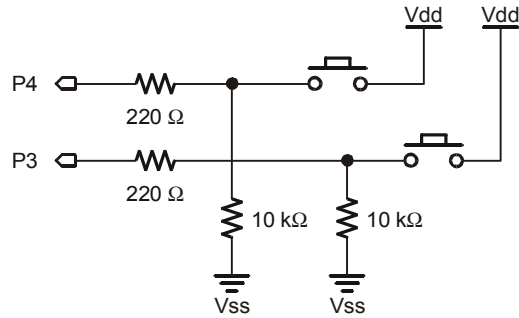
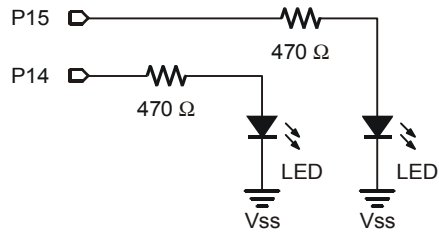



Figura 3-12
Schema Elettrico:
Due Tasti e due LED

Collegare i fili con i pallini: Nella Figura 3-12 ci sono tre posti dove i fili si intersecano, ma solamente pallini. I fili vanno collegati solamente se c'è un pallino sull'intersezione. Il filo che collega il tasto di P4 alla Resistenza da 10 kΩ non è collegato al circuito del tasto di P3 e questa è la ragione del perché non c'è un pallino su quella intersezione.

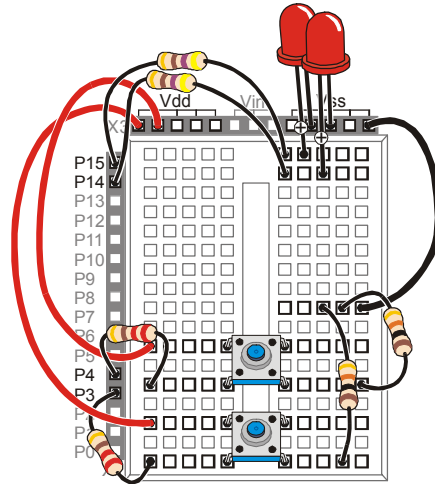



Figura 3-13
Schema di cablaggio:
due tasti e due LED

Programmare il controllo del Tasto

Nell'Esercizio precedente, avete sperimentato come prendere decisioni usando l'istruzione **`IF…THEN…ELSE`**. C'è anche una cosa come l'istruzione **`IF…ELSEIF…ELSE`**. Funziona alla grande per decidere quale LED far lampeggiare. Il prossimo programma mostra come funziona.

Programma di Esempio: TastoControlOfTwoLeds.bs2

- √ Digitate TastoControlOfTwoLeds.bs2 nell'Editor del BASIC Stamp e lanciatelo.
- √ Verificate che il LED nel circuito collegato a P14 lampeggi mentre il tasto collegato a P3 viene mantenuto premuto.
- √ Controllate anche che il LED del circuito collegato a P15 lampeggi quando il tasto nel circuito collegato a P4 viene tenuto premuto.

```
' Che cosa è un Microcontrollore- TastoControlOfTwoLeds.bs2
' Fa lampeggiare il LED P14 se viene premuto il tasto P3, e fa lampeggiare il
' LED P15 se è premuto il tasto P4.

' {$STAMP BS2}
' {$PBASIC 2.5}

DO

  DEBUG HOME
  DEBUG ? IN4
```

```
  DEBUG ? IN3

  IF (IN3 = 1) THEN
    HIGH 14
    PAUSE 50

  ELSEIF (IN4 = 1) THEN
    HIGH 15
    PAUSE 50

  ELSE
    PAUSE 50

  ENDIF

  LOW 14
  LOW 15

  PAUSE 50

LOOP
```

Come funziona TastoControlOfTwoLeds.bs2

Se la visualizzazione di **IN3** e **IN4** scorre verso il basso nel Terminale di Debug come facevano nell'esempio precedente, sarà difficile da leggere. Un modo per risolvere il problema è di mandare sempre il cursore nella posizione in alto a sinistra nella finestra del Terminale di Debug usando il formattatore **HOME**:

```
DEBUG HOME
```

Inviando il cursore nella posizione iniziale ad ogni iterazione di **DO…LOOP**, i comandi:

```
DEBUG ? IN4
DEBUG ? IN3
```

Visualizzeranno i valori di **IN4** e **IN3** ogni volta nella stessa zona della finestra del Terminale di Debug.

In questo programma la parola chiave **DO** inizia il ciclo:

```
DO
```

Questi comandi nell'istruzione **IF** sono gli stessi di quelli del programma esempio dell'Esercizio precedente:

```
IF (IN3 = 1) THEN
  HIGH 14
```

```
    PAUSE 50
```

Questo è il punto dove la parola chiave **ELSEIF** è di aiuto. se **IN3** non è 1, ma **IN4** è 1, vogliamo accendere il LED collegato a P15 invece di quello collegato a P14.

```
ELSEIF (IN4 = 1) THEN
  HIGH 15
  PAUSE 50
```

Se nessuna condizione è vera, noi vogliamo che comunque sia fatta una pausa di 50 ms senza cambiare lo stato dei LED.

```
ELSE
  PAUSE 50
```

Quando avete inserito tutte le decisioni, non dimenticate l'istruzione **ENDIF**.

```
ENDIF
```

È tempo di spengere i LED e fare di nuovo una pausa. Potete provare a decidere quale LED avete acceso e spengerlo. I comandi PBASIC vengono eseguiti molto in fretta, così perché non spengerli semplicemente e smetterla di prendere altre decisioni?

```
LOW 14
LOW 15

PAUSE 50
```

L'istruzione **LOOP** invia il programma di nuovo indietro all'istruzione **DO**, ed il procedimento di controllare i tasti e cambiare lo stato dei LED ricomincia.

```
LOOP
```

Il Vostro Turno – Che cosa ne dite di premere ambedue i Tasti?

Il programma esempio ha un difetto. Provate a premere insieme ambedue i tasti e vedrete il difetto. Vi aspettate di vedere tutti e due i LED lampeggiare, ma non succede perché in un'istruzione **IF...ELSEIF…**solamente un blocco di codice di **ELSE** viene eseguito prima di andare all'istruzione **ENDIF**.

Qui potete vedere come risolvere questo problema:

- √ Salvare TastoControlOfTwoLeds.bs2 con un nuovo nome.
- √ Sostituite questa istruzione **IF** e blocco di codice:

```
IF (IN3 = 1) THEN
```

```
    HIGH 14
    PAUSE 50
```

Con questa istruzione **`IF...ELSEIF`**:

```
  IF (IN3 = 1) AND (IN4 = 1) THEN
    HIGH 14
    HIGH 15
    PAUSE 50

  ELSEIF (IN3 = 1) THEN
    HIGH 14
    PAUSE 50
```

Un blocco di codice è un gruppo di comandi. L'istruzione **`IF`** usata sopra ha un blocco di codice con tre comandi (**`HIGH`**, **`HIGH`**, e **`PAUSE`**). L'istruzione **`ELSEIF`** ha un blocco di codice con due comandi (**`HIGH`**, **`PAUSE`**).

√ Attivate il vostro programma modificato e verificate se controlla ambedue i tasti e circuiti LED come vi sareste aspettati.

La parola chiave AND può essere usata in un'istruzione **`IF…THEN`** per verificare se più di una condizione è vera. Tutte le condizioni con **AND** devono essere vere perché l'istruzione **`IF`** sia vera.

Anche la parola chiave `OR` può essere usata per controllare se almeno una delle condizioni è vera.

Potete anche modificare il programma così che il LED che lampeggia, stia acceso per una quantità di tempo diversa. Per esempio, potete ridurre a 10 l'argomento ***`Duration`*** di **`PAUSE`** per ambedue i tasti, aumentare a 100 **`PAUSE`** per il LED P14, ed aumentare a 200 anche il LED P15.

√ Modificate i comandi **`PAUSE`** nelle istruzioni **`IF`** e nelle due **`ELSEI`** spiegato.
√ Lanciate il programma modificato.
√ Osservate la differenza di comportamento di ciascun LED.

ESERCIZIO #5: MISURATORE DEL TEMPO DI REAZIONE

In una società di video giochi, voi siete il progettista di sistemi a microcontrollore. Il dipartimento commercializzazione, si raccomanda che nel prossimo video gioco sia

inserito un circuito che misuri il tempo di reazione dei giocatori. E quindi il vostro prossimo compito è sviluppare un prototipo reale del misuratore del tempo di reazione.

La soluzione che costruirete e collauderete in questo Esercizio è un esempio di come risolvere questo problema, ma non è certo l'unica soluzione. Prima di continuare, fermatevi un momento a pensare come vorreste progettare il vostro misuratore del tempo di reazione.

Componenti del Misuratore del Tempo di Reazione

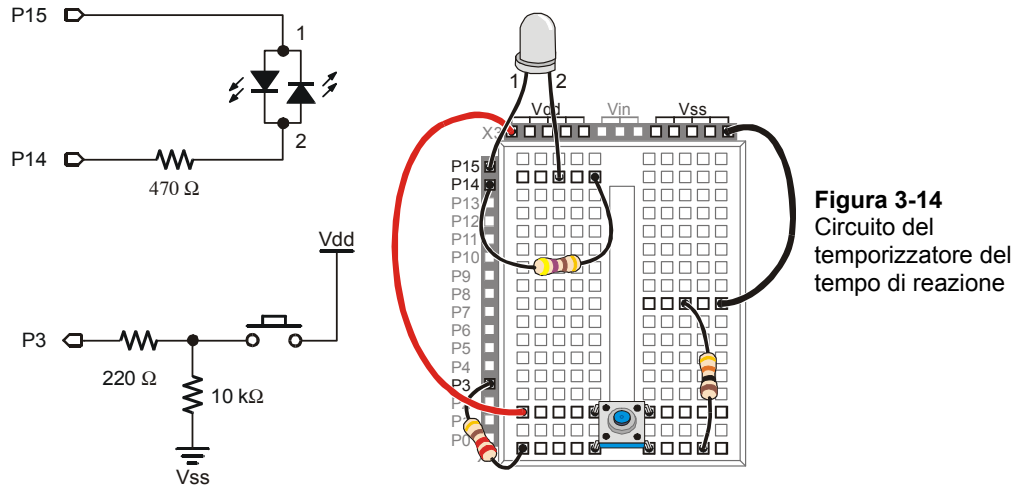
(1) LED – bicolore
(1) Resistenza – 470 Ω (giallo-viola-marrone)

(1) Tasto – normalmente aperto
(1) Resistenza – 10 kΩ (marrone-nero-arancio)
(1) Resistenza – 220 Ω (rosso-rosso-marrone)
(2) Ponticelli

Costruzione del circuito del misuratore del tempo di reazione

La Figura 3-14 mostra lo schema elettrico ed il cablaggio di un circuito che può essere usato con il BASIC Stamp per costruire un gioco per la misura del tempo di reazione.

√ Costruite il circuito mostrato in Figura 3-14.
√ Avviate TestBiColorLED.bs2 del Capitolo #2, Esercizio #5 per provare il circuito del LED bicolore ed assicuratevi che il cablaggio sia corretto.
√ Se avete ricostruito per questo Esercizio il circuito del tasto, lanciate ReadTastoState.bs2 dall'Esercizio #2 in questo Capitolo per assicurarvi che il vostro tasto funzioni correttamente.

Figura 3-14
Circuito del temporizzatore del tempo di reazione

Programmare il Temporizzatore del Tempo di Reazione

Il prossimo programma esempio terrà spento il LED bicolore fino a che il giocatore prema e tenga premuto il tasto. Quando il tasto è tenuto premuto, il LED si accenderà di rosso per un breve periodo. Quando diventa verde, il giocatore deve rilasciare il tasto nel più breve tempo possibile. Il tempo trascorso tra quando il LED è diventato verde ed il rilascio del tasto viene usato per misurare il tempo di reazione.

Il programma esempio dimostra anche come funzionano la rilevazione ed il conteggio. La rilevazione è il procedimento di controllo continuo molto veloce di qualche cosa per vedere se ha cambiato di stato. Il conteggio è il processo di aggiungere un numero ad una variabile ogni volta qualche cosa succede (o non succede). In questo programma, il BASIC Stamp controllerà il tasto dal momento che il LED diventa verde fino al rilascio del tasto. Attenderà 1/1000 di secondo usando il comando **`PAUSE 1`**. Ogni volta che controlla, ed il tasto non è ancora stato rilasciato, aggiungerà 1 alla variabile di conteggio chiamata **`timecounter`**. Quando il tasto viene rilasciato, il programma smette di controllare ed invia un messaggio al Terminale di Debug che visualizza il valore della variabile **`timecounter`**.

Programma di Esempio: ReactionTimer.bs2

√ Digitate e lanciate ReactionTimer.bs2.
√ Seguite gli avvisi sul Terminale di Debug (vedere la Figura 3-15).

Figura 3-15
Istruzioni del Terminale di Debug per il gioco Tempo di Reazione.

```
' Che cosa è un Microcontrollore- ReactionTimer.bs2
' Controlla il Tempo di Reazione con un Tasto ed un LED bicolore.

' {$STAMP BS2}
' {$PBASIC 2.5}

timeCounter    VAR     Word  ' Dichiara la variabile per memorizzare il tempo.

DEBUG "Premere e tenere premuto il Tasto.", CR,      ' Visualizza le istruzioni
      "Fino a che il Led diventa rosso.", CR, CR,    ' del gioco di reazione.
      "Quando il LED diventa verde,", CR,
      "Rilasciare il tasto più velocemente possibile.", CR, CR

DO                                        ' Inizia il ciclo principale.

  DO                                      ' Il ciclo annidato si ripete...
  LOOP UNTIL IN3 = 1                      ' fino a che viene premuto il tasto.

  LOW 14                                  ' Il LED Bicolore acceso rosso.
  HIGH 15

  PAUSE 1000                              ' Ritardo di 1 secondo.

  HIGH 14                                 ' Il LED Bicolore acceso verde.
  LOW 15

  timeCounter = 0                         ' Imposta il contatore del tempo a 0.
```

```
  DO                                    ' Il ciclo annidato conta il tempo...

    PAUSE 1
    timeCounter = timeCounter + 1

  LOOP UNTIL IN3 = 0                    ' Fino al rilascio del tasto.

  LOW 14                                ' LED Bicolore spento.

  DEBUG "Il vostro tempo è stato ", DEC timeCounter, ' Visualizza il tempo
        " ms.", CR, CR,                              ' trascorso.
        "Per giocare ancora, Premere ", CR,          ' Visualizza ancora le
        "e tenere premuto ancora il tasto.", CR, CR  ' istruzioni di gioco.


LOOP                                 ' Vai di nuovo
                                     '"All''inizio del ciclo principale".
```

Come Funziona ReactionTimer.bs2

Dal momento che il programma dovrà tenere traccia del numero di volte che il tasto è stato controllato, viene dichiarata una variabile chiamata **`timeCounter`**.

```
timeCounter VAR Word' Dichiara la variabile per memorizzare il tempo.
```

Le Variabili inizializzate a zero: quando una variabile viene dichiarata nel PBASIC, il suo valore viene posto automaticamente a zero fino a che un comando la imposta ad un nuovo valore.

I comandi **`DEBUG`** contengono le istruzioni di gioco per il giocatore.

```
DEBUG " e tenere premuto il Tasto.", CR,
      " Fino a che il Led diventa rosso..", CR, CR,
      " Quando il LED diventa verde,", CR,
      " Rilasciare il tasto più velocemente possibile.", CR, CR
```

`Le istruzioni DO…LOOP` possono essere annidate. In altre parole, potete mettere un ciclo **`DO…LOOP`** dentro un altro.

```
DO                              ' Inizia il ciclo principale.

  DO                            ' I cicli annidati si ripetono...
  LOOP UNTIL IN3 = 1            ' Fino alla pressione del tasto.

  ' Il resto del programma era quì.
```

```
LOOP               ' Ritorno "all’Inizio del Ciclo Principale".
```

Il ciclo **DO…LOOP** più interno merita un approfondimento. Un ciclo **DO…LOOP** può usare una condizione per decidere se interrompere o no il ciclo ed andare ai comandi successivi. Questo ciclo **DO…LOOP** si ripeterà fino a che il tasto non è premuto (**IN3 = 0**). Il ciclo **DO…LOOP** sarà eseguito fino a che **IN3 = 1**. Quindi, il programma va al comando dopo l’istruzione **LOOP UNTIL**. Questo è un esempio di controllo di stato. Il ciclo **DO…LOOP UNTIL** controlla lo stato del tasto fino a quando viene premuto.

```
DO                          ' I cicli annidati si ripetono...
LOOP UNTIL IN3 = 1          ' Fino alla pressione del tasto.
```

I comandi che vengono immediatamente dopo il ciclo **LOOP UNTIL** fanno le cose seguenti: accensione del LED bicolore rosso, ritardo di un secondo, accensione del LED bicolore verde.

```
LOW 14                       ' Bi-color LED rosso.
HIGH 15

PAUSE 1000                   ' Ritardo 1 secondo.

HIGH 14                      ' Bi-color LED verde.
LOW 15
```

Appena il LED bicolore diventa verde, è tempo di cominciare il conteggio per sapere quanto tempo ha impiegato il giocatore a rilasciare il tasto. La variabile **timeCounter** viene impostata a zero, quindi parte un altro ciclo **DO…LOOP** con una condizione **UNTIL**. Il ciclo si ripete fino a che il giocatore rilascia il tasto (**IN3 = 0**). Ad ogni ciclo, il BASIC Stamp ritarda 1 ms usando **PAUSE 1**, ed inoltre aggiunge 1 al valore della variabile **timeCounter**.

```
timeCounter = 0        ' Imposta timeCounter a zero.

DO                     ' Il ciclo annidato, conta il tempo...

  PAUSE 1
  timeCounter = timeCounter + 1

LOOP UNTIL IN3 = 0    ' fino a che il tasto viene rilasciato.
```

Il LED bicolore si spenge.

```
LOW 14
```

Il risultato viene visualizzato nel Terminale di Debug.

```
DEBUG "Il vostro tempo è stato ", DEC timeCounter,
      " ms.", CR, CR,
      "Per giocare di nuovo, ", CR,
      "premere il tasto.", CR, CR
```

L'ultima istruzione nel programma è **`LOOP`**, che invia il programma indietro alla prima istruzione **`DO`**.

Il Vostro Turno – Revisione del Progetto

Il dipartimento di commercializzazione ha dato il vostro prototipo ad alcuni giocatori-collaudatori. Quando i collaudi sono conclusi, il dipartimento di commercializzazione vi invia un elenco contenente tre problemi che devono essere risolti prima del definitivo inserimento del vostro prototipo nel video gioco.

√ Salvate ReactionTimer.bs2 con un nuovo nome, come per esempio ReactionTimerYourTurn.bs2.

Vengono di seguito discussi, l'elenco dei problemi e la loro risoluzione.

Problema 1

Quando un giocatore tiene premuto il tasto per 30 secondi, il suo punteggio è attualmente 14000 ms, una misura di 14 secondi. Questa imprecisione deve essere rimossa!

È risultato che l'esecuzione stessa del ciclo per aggiungere 1 alla variabile **`timeCounter`** duri circa 1 ms senza il comando **`PAUSE 1`**. questo fenomeno viene chiamato tempo di esecuzione, ed è il tempo che il BASIC Stamp impiega per eseguire i comandi. Un semplice rimedio per aumentare l'accuratezza della misura, è semplicemente escludere il comando **`PAUSE 1`** cancellandolo o anteponendo un apostrofo.

```
' PAUSE 1
```

√ Provate ad apostrofare PAUSE 1 e controllate come è cambiata l'accuratezza della misura.

Invece di apostrofare il comando di ritardo, un altro modo di risolvere il problema è di moltiplicare il vostro risultato per due:

```
timeCounter = timeCounter * 2       ' <- Aggiungete questo
```

```
DEBUG "Il vostro tempo era ", DEC timeCounter, " ms.", CR, CR
```

√ Riattivate il comando **PAUSE** cancellando l'apostrofo, e provate la soluzione con la moltiplicazione.

3

Per aumentare la Precisione, potete usare l'operatore */ per moltiplicare per un valore frazionario che rende la vostra risposta più precisa. L'operatore */ non è difficile da usare; qui è mostrato come usarlo:

1) Mettete il valore o la variabile che volete moltiplicare per un numero frazionario prima dell'operatore */.
2) Moltiplicate il valore frazionario che volete usare per 256.
3) Arrotondatelo alla prima cifra a sinistra del punto decimale.
4) Mettete quel valore dopo l'operatore */.

Esempio: Diciamo di voler moltiplicare la variabile timeCounter per 3,69.

1) Iniziamo mettendo timeCounter a sinistra dell'operatore */:

   timeCounter = timeCounter */

2) Moltiplichiamo il vostro valore frazionario per 256: 3.69 X 256 = 944.64.
3) Arrotondiamo: 944.64 ≈ 945.
4) Mettiamo questo valore a destra dell'operatore */:

   timeCounter = timeCounter */ 945

Problema 2

I giocatori presto si accorgono che il ritardo da rosso a verde è 1 secondo. Dopo aver giocato diverse volte, indovinano meglio quando lasciare il tasto ed il loro punteggio non rispecchia più il loro reale tempo di reazione.

Il BASIC Stamp ha un comando **RANDOM**. Di seguito è spiegato come modificare il vostro codice per ottenere un numero casuale:

√ All'inizio del vostro codice, aggiungete una dichiarazione per una nuova variabile chiamata **value**, ed impostatela a 23. Il valore 23 viene chiamato seme perché dà origine la sequenza di generazione del numero pseudo casuale.

```
timeCounter VAR WORD
value VAR BYTE                          ' <- Aggiungete questo
value = 23                              ' <- Aggiungete questo
```

√ Immediatamente prima del comando pause, usate il comando **RANDOM** per dare a **value** un nuovo valore casuale dalla sequenza di generazione dei numeri pseudo casuali che è cominciata con 23.

```
RANDOM value                       ' <- Aggiungete questo.
DEBUG "Delay time ", ? 1000 + value, CR  ' <- Aggiungete
                                         '    questo.
```

√ Modificate il comando **PAUSE** così che il valore casuale sia aggiunto a 1000 (per un secondo) nell'argomento ***Duration*** del comando **PAUSE**.

```
PAUSE 1000 + value                 ' <- Modificate questo.
```

Che cos'è un algoritmo? Un algoritmo è una sequenza di operazioni matematiche.

Che cosa significa pseudo casuale? Pseudo casuale significa che sembra casuale, ma in realtà non lo è. Ogni volta che il programma riparte da zero, avrete la stessa sequenza di valori.

Che cos'è un seme? Un seme è un valore che viene usato per iniziare una sequenza di generazione di numeri pseudo casuali. Se usaste un valore diverso (cambiate il valore 23 con qualsiasi altro numero), otterreste una diversa sequenza di numeri pseudo casuali.

Problema 3

Un giocatore che rilasciasse il tasto prima che il LED da rosso diventi verde, otterrebbe un tempo di reazione irragionevole (1 ms). Il vostro microcontrollore si deve accorgere se il giocatore sta imbrogliando.

Il codice pseudo casuale alla fine dell'esercizio Esercizio #3 in questo Capitolo. Di seguito ci sono alcuni esempi di pseudo codice per aiutarvi ad applicare l'istruzione **IF…THEN…ELSE** per risolvere il problema.

- *Se il valore di timeCounter è uguale a 1*
    - *Visualizza un messaggio per dire al giocatore che deve attendere che il LED diventi verde prima di rilasciare il tasto.*
- *Inoltre, (se il valore del contatore è maggiore di 1)*
    - *Visualizza il valore di timeCounter in ms (esattamente come in ReactionTimer.bs2).*

√ Modificate il vostro programma implementando questo pseudo codice in PBASIC per risolvere il problema dei giocatori imbroglioni.

SOMMARIO

Questo Capitolo ha introdotto il tasto e i circuiti comunemente usati per gestire i tasti. Questo Capitolo ha anche discusso di come costruire e collaudare un circuito di tasto e di come usare il BASIC Stamp per leggere lo stato di uno o più tasti. Il BASIC Stamp è stato programmato per prendere decisioni basate sullo stato del/dei tasti e questa informazione è stata usata per controllare dei LED. Usando questi concetti è stato costruito un gioco per misurare il tempo di reazione. In aggiunta al controllo dei LED, il BASIC Stamp è stato programmato per sentire lo stato di un tasto e fare misure di tempo.

È stata presentata la lettura dello stato di un tasto usando le variabili specifiche per l'I/O insite nel BASIC Stamp (**`IN3`**, **`IN4`**, etc.). Sono stati inoltre insegnati i modi di prendere decisioni basate su questi valori tramite l'uso delle istruzioni **`IF…THEN…ELSE`**, anche le istruzioni **`IF…ELSEIF…ELSE`**, e i blocchi di codice sono state spiegate. Si è parlato della valutazione di più condizioni, e degli operatori logici **`AND`** ed **`OR`**. È stata trattata l'aggiunta di una condizione ad un ciclo **`DO…LOOP`** usando la parola chiave **`UNTIL`** insieme con la nidificazione di blocchi di codice nel ciclo **`DO…LOOP`**.

Domande

1. Quale differenza c'è fra l'invio e la ricezione dei segnali **`HIGH`** e **`LOW`** usando il BASIC Stamp?
2. Che cosa significa normalmente aperto?
3. Quale sono i segni di avvertimento che indicano che avete commesso un errore serio nel cablaggio del vostro circuito?
4. Che cosa avviene ai terminali di un tasto normalmente aperto quando lo premete?
5. In quale di due stati un circuito di tasto può trovarsi? Quali numeri usa il BASIC Stamp per ciascuno di questi stati?
6. Che cosa vede un pin I/O del BASIC Stamp impostato come ingresso quando un interruttore normalmente aperto che usa una resistenza di pull-down quando viene premuto? Che cosa vede quando il tasto non è premuto?
7. Quando scorre corrente attraverso un circuito con un tasto normalmente aperto?
8. Qual è il valore di **`IN3`** quando un tasto lo collega a Vdd? Qual è il valore di **`IN3`** quando il tasto lo collega a Vss?
9. Che cosa fa il comando **`DEBUG ? IN3`**?
10. Qual è la differenza tra una resistenza di pull-up ed una Resistenza di pull-down?

11. Se modificate un circuito di tasto che aveva una resistenza di pull-down perché abbia una resistenza di pull-up, vi aspettate di vedere cambiamenti nel valore del pin di I/O?
12. Qual è la differenza tra attivo basso e attivo alto?
13. Quale tipo di blocchi di codice possono essere usati per prendere decisioni basate sul valore di uno o più tasti?
14. Che cosa significa annidamento o nidificazione?
15. Che cosa fa il formattatore **HOME** nell'istruzione **DEBUG HOME**?
16. Che cosa significa rilevazione?
17. Che cosa significa supervisione del codice?

Esercizi

1. Disegnate lo schema di un circuito con un tasto normalmente aperto con una Resistenza di pull-down collegata a P5.
2. Disegnate lo schema di un circuito con un tasto normalmente aperto con una Resistenza di pull-up collegata a P5.
3. Spiegare come modificare ReadTastoState.bs2 a pagina 81 così che rilevi lo stato del tasto ogni secondo invece che ogni ¼ di secondo.
4. Spiegare come modificare ReadTastoState.bs2 in modo che rilevi lo stato di un tasto normalmente aperto con una resistenza di pull-up collegato al pin I/O P6.
5. Spiegare come usare un ciclo **FOR…NEXT** in TastoControlledLed.bs2 a pagina 85 in modo che faccia lampeggiare il LED quindici volte prima di controllare che venga premuto di nuovo.
6. Spiegare come modificare TastoControlledLed.bs2 in modo che il LED sia acceso fisso (invece di lampeggiare) quando premete e tenete premuto il tasto.
7. Modificate la versione di TastoControlOfTwoLeds.bs2 della sezione "Il Vostro Turno" in modo che ambedue i LED lampeggino quando nessun tasto viene premuto, e nessun LED lampeggi quando ambedue i tasti siano premuti e tenuti premuti.
8. Spiegare come modificare ReactionTimer.bs2 a pagina 94 in maniera di poter usare **DO WHILE** invece di **LOOP UNTIL**. Scrivete esempi del blocco di codice che dovete cambiare per farlo funzionare. Suggerimento: **LOOP UNTIL** prende le sue decisioni usando una condizione che fa uscire il programma dal ciclo. **DO WHILE** prende le sue decisioni basandosi su una condizione che continua a ripetere il ciclo.

3

Progetti

1. Controllate la vostra risposta all'Esercizio 8 facendo funzionare con un ciclo **`DO WHILE`** ReactionTimer.bs2 invece di un ciclo **`LOOP UNTIL`**.
2. Riferendovi al progetto del semaforo nella sezione Progetti del Capitolo precedente. Alcuni controllori di semafori non lavorano solamente sulla temporizzazione; hanno dei sensori sotto l'asfalto che rilevano se una macchina stia o no aspettando ferma al semaforo rosso. Modificate il controllore semaforico in maniera che la strada nord-sud, sia sempre verde, ma ad una pressione del tasto, inizi la sequenza che permetta alle auto sulla strada est-ovest di passare.
3. Modificate ReactionTimer.bs2 in modo che sia un gioco per due giocatori. Aggiungete un secondo tasto collegato a P4 per il secondo giocatore. SUGGERIMENTI: Usate l'operatore **`AND`** nel vostro ciclo **`DO…LOOP UNTIL`**. Dichiarate due diverse variabili **`timeCounter`**, **`timeCounterA`** e **`timeCounterB`**. Usate il ragionamento **`IF…THEN`** nel ciclo **`DO…LOOP`** che incrementa il contatore. Potete anche prendere a prestito da TastoControlOfTwoLeds.bs2 per aiutarvi a prendere la decisione se incrementare o no uno dei due contatori.

Ulteriori Chiarimenti

Le risorse sotto elencate sono disponibili gratuitamente sul sito web Parallax e sono altresì incluse nel CD Parallax.

"*Manuale del BASIC Stamp*", Users Manual, Version 2.0c, Parallax Inc., 2000

Il *Manuale del BASIC Stamp* ha più esempi che potete provare ed informazioni che spiegano ulteriormente i seguenti argomenti: i formattatori **`DEBUG HOME`** e **`CLS`**, le variabili dei pin di ingresso come **`IN3`**, **`IN4`**, ed il comando **`RANDOM`**.

"*Basic Analog and Digital*", Student Guide, Version 2.0, Parallax Inc., 2003

Basic Analog and Digital Spiega il conteggio binario usando i tasti. Usa inoltre i tasti per introdurre una tecnica per trasmettere numeri da un sistema ad un altro chiamata comunicazione seriale sincrona.

"*Editor del BASIC Stamp Help File*", PBASIC 2.5 Version 2.0 Parallax Inc., 2003
Il file di aiuto del PBASIC 2.5 contiene informazioni sulle condizioni **`WHILE e UNTIL`** usate con il ciclo **`DO…LOOP`**, ed informazioni sull'annidamento e sui blocchi di codice **`IF…THEN…ELSE`**, il che è nuovo sul PBASIC 2.5 e non presente nel Manuale del BASIC Stamp. Potete trovare queste informazioni clickando l'icona libro sulla barra comandi dell'Editor del BASIC Stamp, quindi selezionate PBASIC Reference dal menù nella finestra a sinistra. Questa azione aprirà l'elenco alfabetico di riferimento dei comandi PBASIC nella finestra principale. Clickando su ciascun comando, possono essere trovate informazioni dettagliate.

Capitolo #4: Controllare il Movimento

MOVIMENTO MICROCONTROLLATO

I Microcontrollori assicurano ogni giorno che le cose intorno a voi si muovano come volete. Se avete una stampante a getto di inchiostro, la testina di stampa che và avanti ed indietro viene mossa da un motore passo-passo controllato da un microcontrollore. Le porte automatiche del vostro salumiere che voi attraversate, sono controllate da microcontrollori, e la funzione di espulsione automatica del vostro videoregistratore o del vostro DVDplayer è anch'esso controllato da un microcontrollore.

SEGNALI ON/OFF E MOVIMENTO DI MOTORI

Grosso modo tutti i motori microcontrollati ricevono sequenze di impulsi ON/OFF come quelli che avete inviato ai LED. La differenza è che il microcontrollore deve inviare questi segnali a velocità che sono normalmente superiori alla capacità dell'occhio umano di rilevarli. La temporizzazione ed il numero dei diversi segnali ON/OFF sono diversi da un motore all'altro, ma tutti possono essere controllati da microcontrollori capaci di inviare segnali ON/OFF.

Alcuni di questi motori richiedono molta circuiteria per aiutare i microcontrollori a farli girare. Altri motori richiedono ulteriori parti meccaniche perché siano adattati alle macchine. Di tutti i diversi tipi di motori con cui cominciare, i Servomotori per hobbistica con cui farete gli esperimenti in questo Capitolo, sono probabilmente i più semplici. Come vedrete tra poco, sono facili da controllare con il BASIC Stamp, richiedono pochi o nessun componente aggiuntivi, ed hanno un'uscita meccanica facile da collegare agli oggetti per farli muovere.

ESERCIZIO #1: COLLEGARE E COLLAUDARE I SERVOMOTORI

In questo Esercizio, collegherete un Servomotore ad un alimentatore ed al BASIC Stamp. Verificherete quindi che il Servomotore funzioni correttamente programmando il BASIC Stamp per inviare segnali al Servomotore per controllare la sua posizione.

Conosciamo il Servomotore

La Figura 4-1 mostra un disegno di un Servomotore comunemente usato per hobbistica. Il connettore (1) è usato per collegare il Servomotore ad un'alimentazione (Vdd e Vss) e ad una sorgente di segnale (un pin I/O del BASIC Stamp). Il cavo (2) conduce Vdd, Vss e la

linea segnale dal connettore al Servomotore. La crociera (3) è la parte del Servomotore che sembra una croce. Quando il Servomotore gira, la crociera è la parte mobile che viene controllata dal BASIC Stamp. Il contenitore (4) contiene il circuito di controllo del Servomotore, un motore in corrente continua e gli ingranaggi. Queste parti lavorano insieme per accettare i segnali ON/OFF dal BASIC Stamp e convertirli in posizioni assunte dalla crociera del Servomotore.

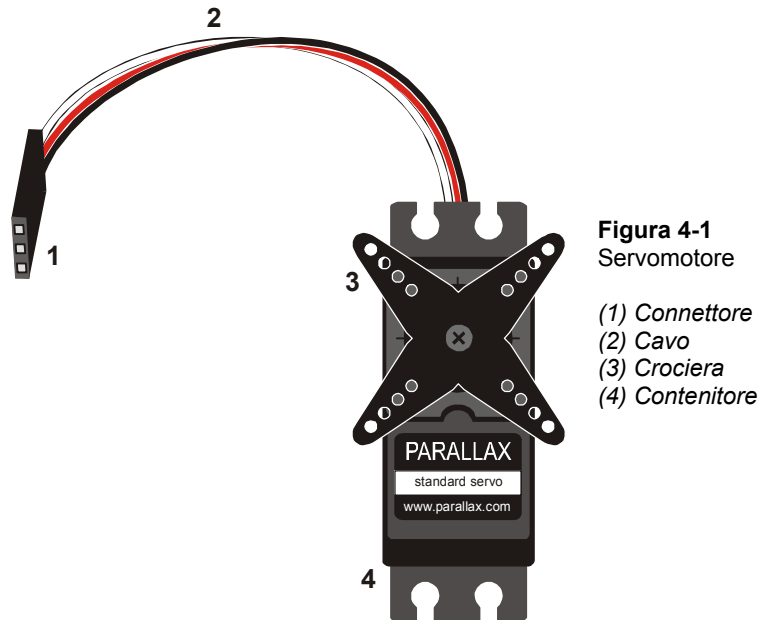



Figura 4-1
Servomotore

(1) Connettore
(2) Cavo
(3) Crociera
(4) Contenitore

Componenti dei circuiti Servomotore e LED

Un circuito LED può essere usato per controllare il segnale che il BASIC Stamp invia al Servomotore. Sappiate comunque che il LED non è essenziale al funzionamento del Servomotore. È presente solamente per aiutarci a vedere che cosa sta succedendo con i segnali di controllo.

(1) Servomotore
(1) Resistenza – 470 Ω (giallo-viola-marrone)
(1) LED – qualsiasi colore

Costruzione dei circuiti Servomotore e LED

È di estrema importanza stare attenti quando si collega un Servomotore al vostro BASIC Stamp. Come collegare il vostro Servomotore dipende se state usando la Board of Education Rev B, Rev C, o la HomeWork Board. Se state usando la Board of Education, ma non siete sicuri della versione in vostro possesso, la Figura 4-2 mostra un esempio dell'etichetta sulla Board of Education Rev B.

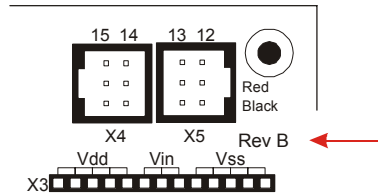



Figura 4-2
Etichetta Rev sulla Board of Education

√ Esaminate l'etichetta sulla scheda e saprete se avete una HomeWork Board o una Board of Education Rev B, o Rev C.

√ Andate alle istruzioni per collegare i Servomotori al BASIC Stamp sulla vostra scheda:

- Pagina 107 – Board of Education Rev C
- Pagina 110 – BASIC Stamp HomeWork Board
- Pagina 113 – Board of Education Rev B

Board of Education Rev C

La Figura 4-3 mostra lo schema del circuito da costruire sulla Board of Education Rev C.

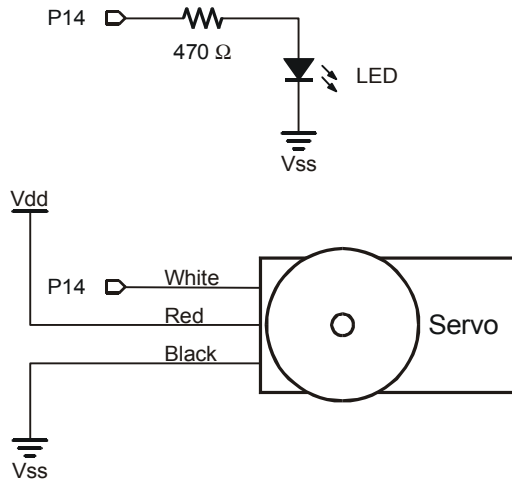



Figura 4-3
Schema del Servomotore e dell'indicatore LED per la Board of Education Rev C

√ Spengete l'alimentazione come mostrato in Figura 4-4.

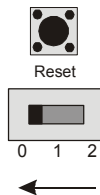



Figura 4-4
Spengere l'alimentazione

La Figura 4-5 mostra i connettori dei Servomotori sulla Board of Education Rev C. i Servomotori dovranno essere inseriti qui. Questa scheda possiede un ponticello che vi dà la possibilità di collegare l'alimentazione dei Servomotori sia a Vin che a Vdd. Il ponticello è il pezzettino rimuovibile di plastica nera rettangolare giusto tra i due connettori per i Servomotori.

√ Inserite il ponticello su Vdd come mostrato in Figura 4-5. Questo significa togliere il ponticello dai pin dove è attualmente, e reinserirlo sui due pin vicino all'etichetta Vdd.

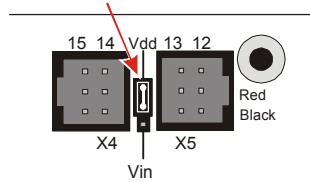



Figura 4-5
Ponticello dei Servomotori impostato a Vdd

Il ponticello vi permette di scegliere l'alimentazione dei Servomotori (Vin o Vdd). Se state usando una pila da 9 V, qualsiasi impostazione funzionerà con un solo Servomotore Standard Parallax. Usate Vin se state usando un pacco batterie da 6 V. Usate Vdd se state usando un alimentatore da parete che si collega ad una presa in corrente alternata. Prima di provare a collegare un alimentatore in corrente continua, assicuratevi che sia compatibile con le specifiche elencate nell'**Error! Reference source not found.**.

√ Costruite il circuito mostrato in Figura 4-6.

√ Assicuratevi di non aver invertito i contatti del Servomotore. I fili bianco, rosso e nero devono essere nello stesso ordine della Figura.

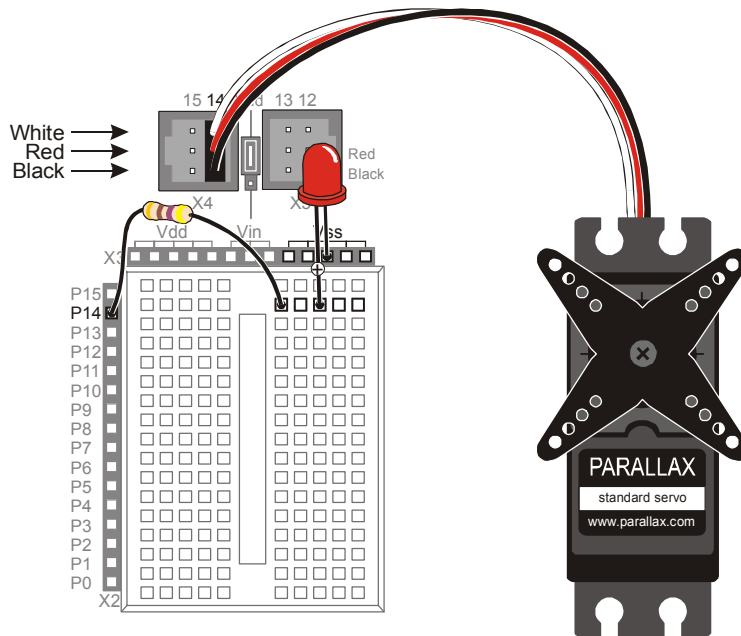



Figura 4-6
Servomotore ed indicatore LED sulla Board of Education Rev C

Fino ad ora, avete usato l'interruttore a tre posizioni nella posizione 1. Ora, lo sposterete nella posizione 2 per alimentare anche i connettori dei Servomotori.

√ Alimentate il connettore dei Servomotori spostando l'interruttore a tre posizioni come mostrato in Figura 4-7. Il vostro Servomotore può fare qualche movimento quando accendete.

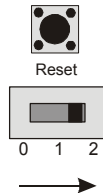



Figura 4-7
Alimentazione della Board of Education e del connettore dei Servomotori

√ Andate alla sezione Programmazione del controllo del Servomotore a pagina 1155.

BASIC Stamp HomeWork Board

Se state collegando il vostro Servomotore alla HomeWork Board, avrete bisogno di questi ulteriori componenti:

(1) Connettore a 3-pin maschio/maschio (mostrato in Figura 4-8).
(4) Ponticelli

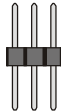



Figura 4-8
Ulteriori componenti per HomeWork Board o Board of Education

(1) Connettore 3-pin maschio/maschio

La Figura 4-9 mostra lo schema elettrico dei circuiti del Servomotore e del LED sulla HomeWork Board. Le istruzioni seguenti vi mostreranno come costruire in sicurezza questo circuito.

AVVISO

Usate solamente pile da 9 V quando un Servomotore è collegato alla BASIC Stamp HomeWork Board. Non usate alcun genere di alimentatore che si colleghi alla rete in corrente alternata. L'uso improprio di questi dispositivi può causare il non funzionamento dell'Esercizio, o perfino il danneggiamento permanente del Servomotore.

Per i migliori risultati, usate pile nuove. Se state usando pile ricaricabili, assicuratevi che siano state ricaricate accuratamente. Dovrebbero anche avere una capacità di 100 mAh (milliampere/ora) o maggiore. Vedere l'**Error! Reference source not found.** per ulteriori informazioni.

4

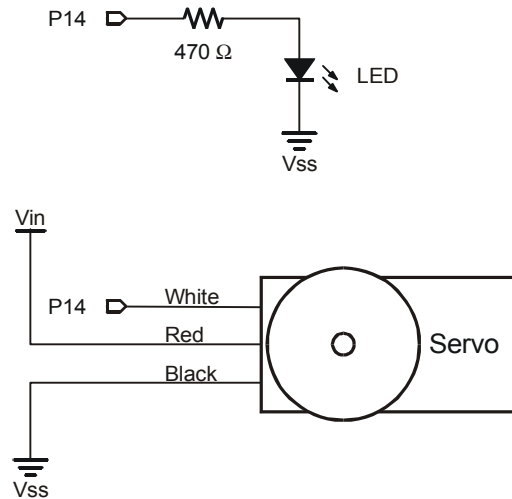



Figura 4-9
Schema elettrico del Servomotore e dell'indicatore a LED sulla HomeWork Board

√ Scollegate la pila da 9 V dalla vostra HomeWork Board.
√ Costruite il circuito dell'indicatore LED ed il connettore del Servomotore mostrato nella Figura 4-10.

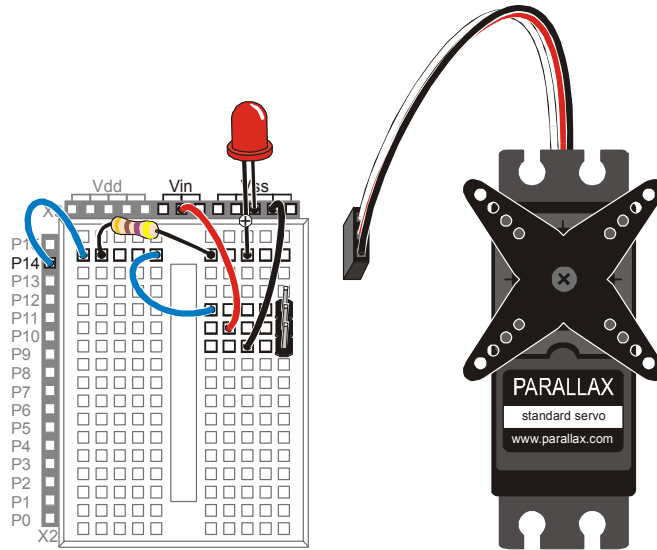



Figura 4-10
Circuito dell'indicatore LED e del Servomotore sulla HomeWork Board

√ Collegate il Servomotore al connettore del Servomotore come mostrato nella Figura 4-11.

√ Assicuratevi che i colori del cavo del Servomotore siano correttamente orientati come i colori del disegno.

√ Ricontrollate il vostro cablaggio.

√ Ricollegate la pila da 9 V alla vostra HomeWork Board. Il Servomotore può muoversi un po' mentre fate il collegamento.

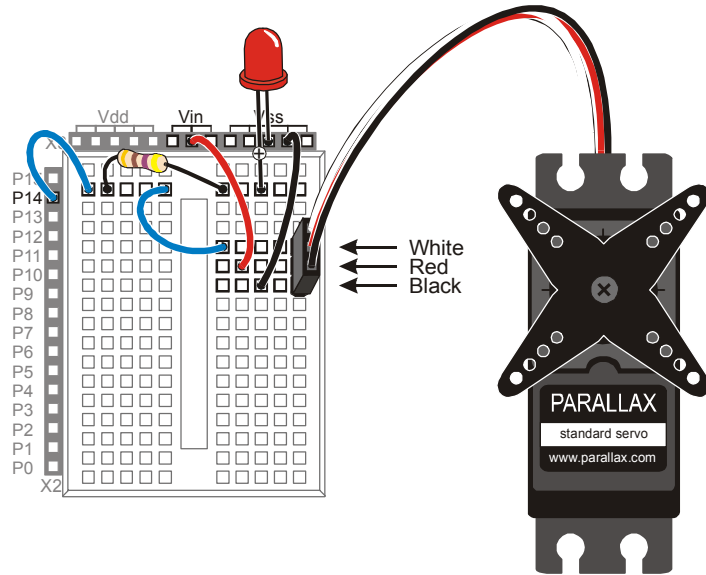



Figura 4-11
Collegamento del Servomotore al connettore per il Servomotore sulla HomeWork Board

√ Andate alla sezione Programmazione del controllo del Servomotore a pagina 1155.

Board of Education Rev B

La Figura 4-12 mostra lo schema elettrico per i circuiti del Servomotore e del LED sulla Board of Education Rev B. le istruzioni che seguono vi guideranno a costruire in maniera sicura questo circuito sulla Board of Education Rev B.

AVVISO

Usate solamente pile da 9 V quando un Servomotore è collegato alla Board of Education Rev B. Non usate alcun genere di alimentatore che si colleghi alla rete in corrente alternata. L'uso improprio di questi dispositivi può causare il non funzionamento dell'Esercizio, o perfino il danneggiamento permanente del Servomotore.

Per i migliori risultati, usate pile nuove. Se state usando pile ricaricabili, assicuratevi che siano state ricaricate accuratamente. Dovrebbero anche avere una capacità di 100 mAh (milliampere/ora) o maggiore. Vedere l'**Error! Reference source not found.** per ulteriori informazioni.

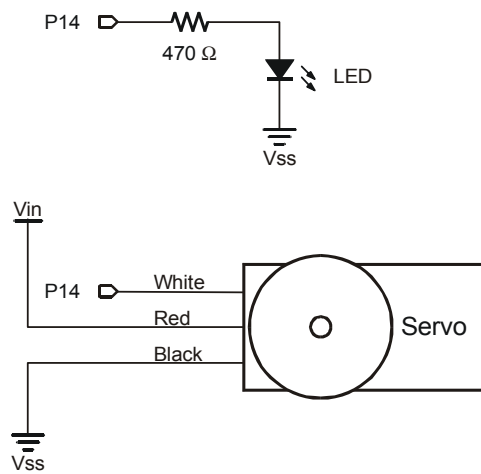



Figura 4-12
Schema elettrico del circuito del Servomotore e dell'indicatore a LED sulla Board of Education Rev B

√ Scollegate la pila o qualsiasi altra alimentazione dalla vostra scheda.
√ Costruite il circuito del LED mostrato in Figura 4-12.
√ Collegate il Servomotore al connettore per il Servomotore come mostrato in Figura 4-13.

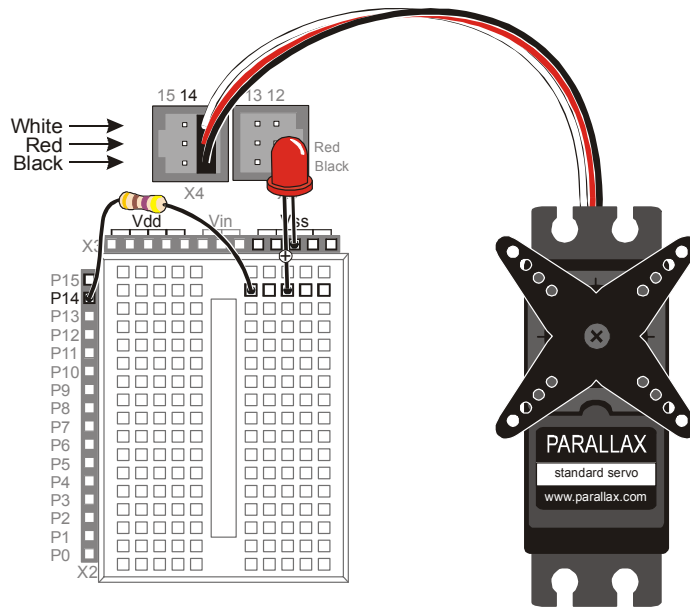



Figura 4-13
Collegamento del Servomotore al connettore per Servomotore sulla Board of Education Rev B.

√ Assicuratevi che i colori del cavo del Servomotore siano allineati correttamente secondo quanto indicato nel disegno.

√ Collegate la pila da 9 V alla Board of Education Rev B. Il Servomotore può muoversi leggermente quando fate la connessione.

Programmazione del controllo del Servomotore

Un Servomotore viene controllato da segnali alti molto brevi. Questi segnali alti, vengono mandati in continuazione ogni 20 ms. i segnali alti hanno una durata qualsiasi tra 1 e 2 ms. Per inviare questo impulso (un segnale alto molto breve) può essere usato il comando **PULSOUT** che usa un pin I/O del BASIC Stamp. Segue la sintassi del comando **PULSOUT**:

PULSOUT *Pin, Duration*

Come con **HIGH** e **LOW**, l'argomento ***Pin*** è un numero che dice al BASIC Stamp da quale pin I/O il segnale viene inviato. L'argomento ***Duration*** non è in millisecondi come nel comando **PAUSE**. Per il BASIC Stamp 2, l'argomento ***Duration*** è il numero di periodi di 2 milionesimi di secondo (µs) di tempo in cui volete che il segnale resti alto.

Un milionesimo di secondo viene chiamato microsecondo. Viene usata la lettera Greca μ al posto di micro e la lettera s viene usata al posto di secondo. Questo torna comodo quando si scrive o si prende nota perché invece di scrivere 2 microsecondi, potete scrivere 2 μs.

Per Ricordare: un millesimo di secondo viene chiamato un millisecondo, e si abbrevia in ms.

Fatti: 1 ms = 1000 μs. In altre parole, ci saranno mille milionesimi di secondo in un millesimo di secondo.

Il prossimo programma esempio userà il comando **PULSOUT** per inviare impulsi che indicheranno al Servomotore come posizionare la sua crociera. Verranno usato i cicli **FOR…NEXT** per inviare un certo numero di impulsi, che faranno posizionare in una determinata posizione il Servomotore per un determinato periodo di tempo.

Programma di Esempio: ServomotoreTest.bs2

ServomotoreTest.bs2 muovere la crociera del Servomotore ad una posizione all'incirca ad ore dieci e mantenere questa posizione per circa tre secondi. Quindi, il programma muoverà la crociera del Servomotore in senso orario, nella posizione ad ore 2 per circa tre secondi. Dopo di che, il Servomotore manterrà la sua "posizione centrale", che è all'incirca ad ore 12, per circa tre secondi.

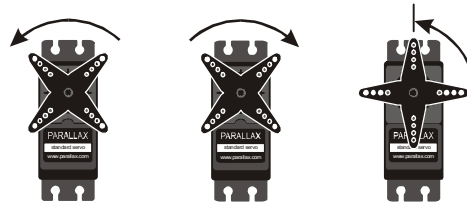



Figura 4-14
Movimento della crociera del Servomotore

Ore 10 (a sinistra)
Ore 2 (al centro)
Ore 12 (a destra)

Che cosa succede se il mio Servomotore è diverso? Esistono molti Servomotori diversi, e molti possono reagire in maniera diversa agli impulsi che TestServomotore.bs2 invia. Il vostro Servomotore può ruotare solamente fino ad ore 11 quindi fino ad ore 1, oppure ad ore 9 e quindi ad ore 3. Può perfino ruotare nella direzione opposta e cominciare in senso orario e poi andare in senso antiorario. Dal momento che il movimento che si osserva è deciso e consistente, torna comodo per questi esercizi. Potete sempre modificare i programmi esempio per far comportare il Servomotore nel modo che volete.

√ Digitate ServomotoreTest.bs2 nell'Editor del BASIC Stamp.
√ Accendete la vostra Board of Education o HomeWork Board.

√ Lanciate il programma.

√ Osservate il Servomotore girare a ciascuno dei tre passi del programma, e annotate dove la crociera sta puntando in quel momento.

√ Rilanciate il programma e verificate che il LED lampeggi fiocamente. Dovrebbe essere più brillante quando il BASIC Stamp invia il segnale "ore 10" e più fioco quando il BASIC Stamp invia il segnale "ore 2". Questo perché nel secondo caso il circuito del LED è acceso per la metà del tempo (1 di 21 ms invece che 2 di 22 ms).


```
' Che cosa è un Microcontrollore- ServomotoreTest.bs2
' Prova il Servomotore con segnali per tre posizioni.

' {$STAMP BS2}
' {$PBASIC 2.5}

counter        VAR     Word

DEBUG "Counterclockwise 10 o'clock", CR

FOR counter = 1 TO 150
  PULSOUT 14, 1000
  PAUSE 20
NEXT

DEBUG "Clockwise 2 o'clock", CR

FOR counter = 1 TO 150
  PULSOUT 14, 500
  PAUSE 20
NEXT

DEBUG "Center 12 o'clock", CR

FOR counter = 1 TO 150
  PULSOUT 14, 750
  PAUSE 20
NEXT

DEBUG "All done."

END
```

Come Funziona ServomotoreTest.bs2

Il primo ciclo **FOR…NEXT** invia 150 impulsi, ciascuno dei quali dura 2.0 ms. Questi impulsi indicano al Servomotore di posizionarsi all'incirca ad ore 10 (pensate di guardare un orologio).

```
FOR counter = 1 TO 150
  PULSOUT 14, 1000
  PAUSE 20
NEXT
```

PULSOUT 14, 1000 invia un impulso che dura 1000 × 2 µs. che è 2000 µs o 2 ms.

La Figura 4-15 si chiama diagramma di temporizzazione. Mostra un disegno dei segnali alto e basso e quanto durano. Il diagramma di temporizzazione non mostra quanti sono gli impulsi inviati, ma vi dà le informazioni sulla durata e l'intervallo degli impulsi. Ciascun impulso (segnale alto) dura 2.0 ms. Ciascun impulso è separato da un segnale basso della durata di 20 ms.

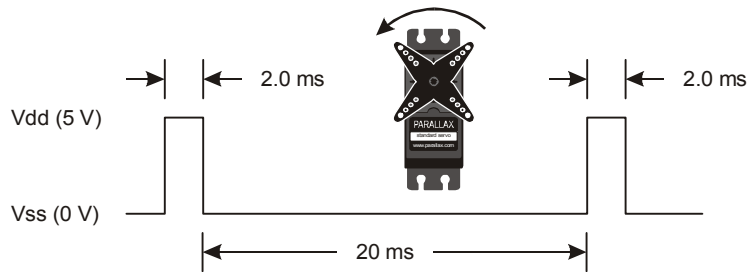



Figura 4-15
Diagramma di temporizzazione per impulsi di 2.0 ms ogni 20 ms

Crociera del Servomotore nella posizione ore 10.

Il secondo ciclo **FOR…NEXT** invia 150 impulse, ma questa volta, ciascun impulso dura soltanto 1.0 ms. Questo dice al Servomotore di posizionarsi ad ore 2 per circa 3.15 secondi.

```
FOR COUNTER = 1 TO 150
  PULSOUT 14, 500
  PAUSE 20
NEXT
```

PULSOUT 14, 500 invia un impulso che dura 500 × 2 µs. che è di 1000 µs o 1 ms.

La Figura 4-15 mostra il diagramma di temporizzazione per questo treno di impulsi. Le pause tra gli impulsi durano ancora 20 ms.

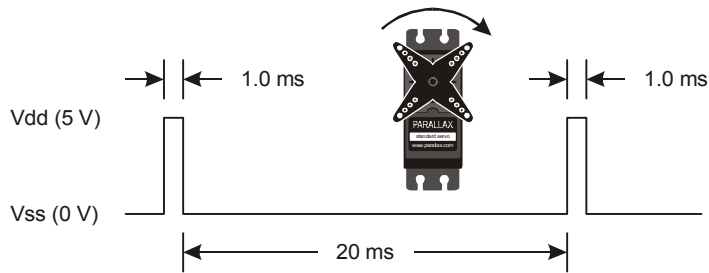



Figura 4-16
Diagramma di temporizzazione per impulsi di 1.0 ms ogni 20 ms

Crociera del Servomotore nella posizione ore 2.

L'ultimo ciclo **`FOR…NEXT`** invia 150 impulsi, ciascuno dei quali dura 1.5 ms. Questo indica al Servomotore di andare nella posizione di centro (ore 12) per circa 3.23 secondi.

```
FOR counter = 1 TO 150
  PULSOUT 14, 750
  PAUSE 20
NEXT
```

`PULSOUT 14, 750` invia un impulso che dura 750 × 2 µs. Equivalenti a 1500 µs o 1.5 ms.

La Figura 4-17 mostra il diagramma di temporizzazione di questi impulsi. Mentre il tempo "basso" è ancora 20 ms, l'impulso "alto" dura ora 1.5 ms.

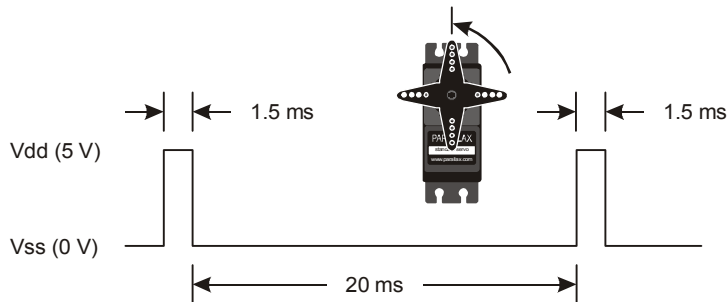



Figura 4-17
Diagramma di temporizzazione per impulsi di 1.5 ms ogni 20 ms

Crociera del Servomotore in posizione ore 12.

Fate un Po' di Calcoli Matematici

Se volete convertire il tempo in millisecondi ad un numero che potete usare per l'argomento ***`Duration`*** per il comando **`PULSOUT`**, usate questa equazione.

$$Duration = number\ of\ ms \times 500$$

Per esempio, se ancora non sapete che per un argomento di **`PULSOUT`** per 1.5 ms il numero è 750, di seguito è spiegato come calcolarlo.

$$\begin{aligned} Duration &= 1.5 \times 500 \\ &= 750 \end{aligned}$$

Potete anche ricavare la ***`Duration`*** di un **`PULSOUT`** sconosciuto usando questa equazione.

$$number\ of\ ms = \frac{Duration}{500} ms$$

Per esempio, se vedete il comando **`PULSOUT 14, 850`**, quanto dura realmente quell'impulso?

$$\begin{aligned} number\ of\ ms &= \frac{850}{500} ms \\ &= 1.7\, ms \end{aligned}$$

Il Vostro Turno – Regolare la posizione ed il tempo di Tenuta

Il parametro che controlla per quanto tempo il Servomotore sta in una data posizione è il numero di volte che il ciclo **`FOR…NEXT`** viene reiterato. Il valore dell'argomento ***`Duration`*** del comando **`PULSOUT`** controlla dove e per quanto gira il Servomotore. È importante sperimentare cambiando questi valori per essere sicuri che funzionino prima di passare al prossimo esperimento.

√ Salvate ServomotoreTest.bs2 con il nome ServomotoreTestYourTurn.bs2.

√ Modificate tutti i cicli **`FOR...NEXT`** in modo che vengano eseguiti la metà delle volte rispetto al programma originale:

```
FOR counter = 1 to 75
```

√ Lanciate il programma modificato e verificate che il Servomotore rimane in ciascuna posizione per la metà del tempo.

√ Modificate tutti i cicli **`FOR...NEXT`** in modo che vengano eseguiti il doppio delle volte rispetto al programma:

```
FOR counter = 1 to 300
```

√ Lanciate il programma modificato e verificate che il Servomotore rimane in ciascuna posizione per il doppio del tempo.

√ Modificate il comando **`PULSOUT`** nel primo ciclo in modo che sia**:**

```
PULSOUT 14,850
```

√ Modificate il comando **PULSOUT** nel secondo ciclo in modo che sia**:**

```
PULSOUT 14,650
```

√ Lanciate il programma modificato e spiegate le differenze di spostamento e di posizionamento.

4

ESERCIZIO #2: CONTROLLARE IL POSIZIONAMENTO CON IL VOSTRO COMPUTER

L'Automazione Industriale spesso implica la comunicazione fra microcontrollori e computer più grandi. I microcontrollori leggono i dati dei sensori e li inviano al computer principale. Il computer principale interpreta ed analizza i dati dei sensori, quindi invia l'informazione di posizione al microcontrollore. Il microcontrollore può quindi aggiornare la velocità di un nastro trasportatore, la posizione di un selettore o di qualche altro meccanismo controllato da un motore.

Potete usare il Terminale di Debug per inviare informazioni dal vostro computer al BASIC Stamp come mostrato in Figura 4-18. Il BASIC Stamp deve essere programmato per "ascoltare" i messaggi che voi gli inviate usando il Terminale di Debug, e deve anche memorizzare i dati che gli inviate in una o più variabili.

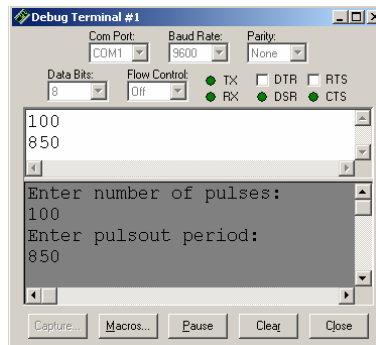



Figura 4-18
Invio di Messaggi al BASIC Stamp

Clickare nel campo bianco sopra alla finestra dei messaggi e digitare il vostro. Una copia del messaggio che avete inserito, apparirà nella finestra di ricezione. Questa copia viene chiamata eco.

In questo Esercizio, programmerete il BASIC Stamp per ricevere due valori dal Terminale di Debug:

1. Il numero degli impulsi da inviare al Servomotore
2. Il valore ***Duration*** usato dal comando **PULSOUT**

Programmerete il BASIC Stamp anche per usare questi valori per controllare il Servomotore.

Componenti e Circuito

Gli stessi dell'Esercizio #1

Programmare il BASIC Stamp per ricevere messaggi dal Terminale di Debug

Programmare il BASIC Stamp per inviare messaggi al Terminale di Debug viene fatto usando il comando **DEBUG**. La programmazione del BASIC Stamp per ricevere messaggi dal Terminale di Debug viene fatta usando il comando **DEBUGIN**. Quando si usa questo comando, dovete anche dichiarare una o più variabili dove il BASIC Stamp possa memorizzare le informazioni che riceve. Di seguito viene mostrato un esempio di variabile che potete dichiarare per far memorizzare il valore al BASIC Stamp:

```
pulses VAR Word
```

In seguito nel programma, potrete usare questa variabile per memorizzare il valore ricevuto dal comando **DEBUGIN**:

```
DEBUGIN DEC pulses
```

Quando il BASIC Stamp riceve un valore numerico dal Terminale di Debug, lo memorizzerà nella variabile **pulses**. Il formattatore **DEC** informa il comando **DEBUGIN** che i caratteri che state inviando, saranno cifre in formato decimale. Appena premete il tasto Enter, il BASIC Stamp memorizzerà le cifre ricevute nella variabile **pulses** come numero decimale, quindi proseguirà con il programma.

Sebbene ciò non sia incluso nel programma esempio, potete includere una riga di codice per verificare che il messaggio sia stato eseguito dal BASIC Stamp.

```
DEBUG CR, "Avete inviato il valore: ", DEC pulses
```

Programma di Esempio: ServomotoreControlWithDebug.bs2

La Figura 4-19 mostra una vista della finestra di trasmissione nel Terminale di Debug. La vostra finestra di trasmissione potrebbe essere più piccola. Per ingrandirla potrete clickare e trascinare il separatore tra le due finestre. Potete digitare le cifre nella finestra di trasmissione per inviarle al BASIC Stamp. In questo esempio, qualcuno ha digitato 100, quindi Enter, poi ha digitato 850. La finestra di ricezione visualizza la scritta

"Digitare il numero degli impulsi:" inviato dal BASIC Stamp. Visualizza anche l'eco del carattere 100 che era stato digitato nella finestra di trasmissione.

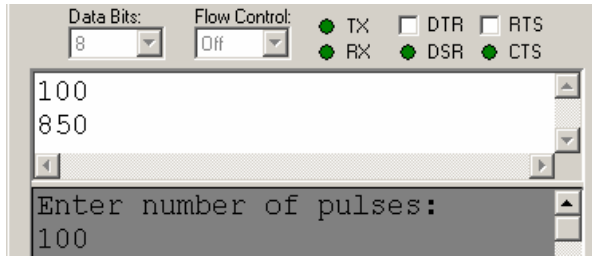



Figura 4-19
Finestre del Terminale di Debug:

← *Finestra di Trasmissione*

← *Separatore delle due Finestre*
← *Finestra di Ricezione*

Eco: quando inviate un messaggio, ed una copia di quel messaggio appare nella vostra Finestra di Ricezione questo si chiama **Eco**. Potete disattivare l'eco clickando nella finestrella di abilitazione mostrata sotto (dovrebbe apparire un segno di spunta). Questo farà cessare la visualizzazione di questi eco.

√ Digitate ServomotoreControlWithDebug.bs2 nell'Editor del BASIC Stamp ed attivatelo.
√ Se la finestra di trasmissione fosse troppo piccola, ridimensionatela usando il mouse per clickare, bloccare, e trascinare in basso il separatore tra le due finestre. Il separatore è mostrato nella Figura 4-19 subito sopra il messaggio: "Digitare il numero degli impulsi:".
√ Clickare nella finestra superiore, di trasmissione, per posizionare il cursore e digitare i messaggi.
√ Quando il Terminale di Debug vi suggerisce di, "Digitare il numero degli impulsi:", digitate il numero 100, quindi premete Enter.
√ Quando il Terminale di Debug vi suggerisce di "Digitate la durata di PULSOUT:" digitate il numero 850, quindi premete Enter.

La `Duration` di PULSOUT deve essere un numero tra 500 e 1000. Se digitate un numero al fuori di questi limiti, il Servomotore può provare a ruotare fuori dai suoi limiti meccanici. Sebbene ciò non danneggi il Servomotore, ne potrebbe accorciare la vita utile.

Il BASIC Stamp mentre invia gli impulsi al Servomotore visualizzerà il messaggio "Il Servomotore sta girando…". Quando avrà finito di inviare impulsi al Servomotore, visualizzerà il messaggio "Fatto" per un secondo. Quindi, vi richiederà di inserire di nuovo il numero degli impulsi. Divertitevi, ma assicuratevi di seguire le direttive contenute nel riquadro Avvertimenti, di restare, per il valore di **PULSOUT** nella gamma tra 500 e 1000.

√ Sperimentate con l'inserimento di altri valori tra 500 e 1000 per il valore dell'argomento ***Duration*** di **PULSOUT** e valori tra 1 e 65534 per il numero di impulsi.

Ci vogliono tra i 40 ed i 45 impulsi per far mantenere la posizione al Servomotore per 1 secondo.

```
' Che cosa è un Microcontrollore- ServomotoreControlWithDebug.bs2
' Invia messaggi al BASIC Stamp per controllare un Servomotore usando il
' Terminale di Debug.

' {$STAMP BS2}
' {$PBASIC 2.5}

counter        Var     Word
pulses         Var     Word
duration       Var     Word

DO

  DEBUG CLS, "Digitare il numero degli impulsi:", CR
  DEBUGIN DEC pulses

  DEBUG "Digitare la durata di Pulsout:", CR
  DEBUGIN DEC duration

  DEBUG "Il Servomotore sta girando...", CR

  FOR counter = 1 TO pulses
    PULSOUT 14, duration
    PAUSE 20
  NEXT

  DEBUG "Fatto"
  PAUSE 1000

LOOP
```

Come funziona ServomotoreControlWithDebug.bs2

In questo programma vengono dichiarate tre variabili **Word**:

```
counter        Var     WORD
pulses         Var     WORD
duration       Var     WORD
```

La variabile **counter** viene dichiarata per l'uso con un ciclo **FOR…NEXT**. Per i dettagli vedere il Capitolo #2, Esercizio #3. Le variabili **pulses** e **duration** sono usate in due modi diversi. Sono ambedue usate per ricevere e memorizzare i valori inviati dal Terminale di Debug. La variabile **pulses** viene anche usata per impostare il numero di iterazioni del ciclo **FOR…NEXT** che invia gli impulsi al Servomotore, e la variabile **duration** viene usata per impostare la durata del comando **PULSOUT**.

Il resto del programma è annidato all'interno di un ciclo **DO…LOOP** senza argomenti **WHILE** o **UNTIL** in modo che i comandi vengano eseguiti all'infinito.

```
DO
  ' il resto del programma non è mostrato.
LOOP
```

Il comando **DEBUG** viene usato per inviarvi ("l'utente" del software) un messaggio per l'inserimento del numero degli impulsi. Quindi, il comando **DEBUGIN** attende che voi inseriate le cifre e premiate il tasto Enter sulla vostra tastiera. Le cifre che inserite vengono convertite in un valore che viene memorizzato nella variabile **pulses**. Questo procedimento viene ripetuto con un secondo comando **DEBUG** e **DEBUGIN** che caricano il valore anche nella variabile **duration.**

```
DEBUG CLS, "Digitare il numero degli impulsi:", CR
DEBUGIN DEC pulses

DEBUG "Digitare la durata degli impulsi:", CR
DEBUGIN DEC duration
```

Dopo l'inserimento del secondo valore, è utile visualizzare un messaggio mentre il Servomotore sta girando cosi che non tentiate di inserire un ulteriore valore:

```
DEBUG "Il Servomotore sta girando...", CR
```

Mentre il Servomotore sta girando, potete provare delicatamente a spostare la crociera del Servomotore dalla posizione che sta mantenendo. Il Servomotore dovrebbe resistere ad una piccola pressione applicata alla crociera.

FOR Counter = StartValue TO EndValue {STEP StepValue}…NEXT

Questa è la sintassi del ciclo **FOR…NEXT** tratta dal Manuale del BASIC Stamp. Essa mostra che sono necessari i valori per gli argomenti ***Counter***, ***StartValue*** ed ***EndValue*** per controllare quante volte il ciclo si ripete. C'è anche un argomento opzionale ***StepValue*** se volete aggiungere un numero diverso da 1 al valore di ***Counter*** ad ogni iterazione del ciclo.

Come negli esempi precedenti, la variabile **counter** viene usata come indice per il ciclo **FOR…NEXT**. Fino a questo esempio, tutti i cicli **FOR…NEXT** hanno usato costanti come 10 o 150 come valori **EndValue**. In questo ciclo **FOR…NEXT**, il valore della variabile degli impulsi viene usata per impostare il parametro **EndValue** del ciclo **FOR…NEXT**. Questo, a sua volta, imposta quanti impulsi vengono inviati al Servomotore. Il risultato finale è che la variabile **pulses** controlla per quanto tempo il Servomotore mantiene una data posizione. Fino ad ora, inoltre, per l'argomento ***Duration*** del comando **PULSOUT** sono stati usati valori costanti come 500, 750, e 1000. Esaminate attentamente questo ciclo **FOR…NEXT** per vedere dove e come vengono usate queste variabili:

```
FOR counter = 1 to pulses
  PULSOUT 14, duration
  PAUSE 20
NEXT
```

Prendetevi del tempo per capire questo ciclo FOR…NEXT. Questa è uno dei primi esempi delle cose mirabili che potete fare con le variabili, i comandi e gli argomenti in PBASIC, pone inoltre l'attenzione sull'utilità di un microcontrollore programmabile come il BASIC Stamp.

Il Vostro Turno – Impostare i Limiti via Software.

Il programma esempio non evita che voi o qualcun altro inserisca una ***duration*** di **PULSOUT** di per esempio 1500, che non fa del bene al Servomotore. Questo è un problema che deve essere risolto se dovete inserire questo sistema in un prodotto.

Immaginiamo che questo sistema di controllo computerizzato del Servomotore sia stato sviluppato per controllare a distanza una porta. Forse una guardia della sicurezza userà il sistema per aprire una porta scorrevole guardandola da una telecamera. Può darsi che un laboratorio scolastico, lo usi per controllare le porte di un labirinto in cui dei topi vadano

in cerca di cibo. O forse un artigliere lo userà per puntare il cannone su un obiettivo particolare. Se state progettando il prodotto per essere usato da qualcun altro, l'ultima cosa che desiderate è di dargli la possibilità (guardia della sicurezza, laboratorio scolastico, artigliere) di inserire il numero sbagliato danneggiando così l'apparato.

Per risolvere questo problema, provate i passi seguenti:

√ Salvate il programma esempio ServomotoreControlWithDebug.bs2 con il nuovo nome di ServomotoreControlWithDebugYourTurn.bs2.

√ Sostituite questi due comandi:

```
DEBUG "Digitare la durata degli impulsi:", CR
DEBUGIN DEC duration
```

Con questo blocco di codice:

```
DO
 DEBUG " Digitare la durata degli impulsi:", CR
  DEBUGIN DEC duration
   IF duration < 500 THEN
    DEBUG "Il valore della durata deve essere superiore a 499", CR
     PAUSE 1000
   ENDIF
    IF duration > 1000 THEN
      DEBUG " Il valore della durata deve essere inferiore a 1001", CR
       PAUSE 1000
     ENDIF
   LOOP UNTIL duration > 499 AND duration < 1001
```

√ Salvate il programma.

√ Lanciate il programma e verificate che respinga i valori al di fuori della gamma appropriata per il Servomotore.

ESERCIZIO #3: CONVERTIRE LA POSIZIONE IN MOVIMENTO

In quest'Esercizio, programmerete il Servomotore per cambiare la posizione con differenti rapporti. Cambiando la posizione con differenti rapporti, farete ruotare la crociera del Servomotore a velocità diverse. Potete usare questa tecnica per controllare il movimento del Servomotore invece della posizione.

Programmare un rapporto di cambio della Posizione

Potete usare un ciclo **FOR…NEXT** come il seguente per far assumere al Servomotore tutte le posizioni della sua gamma:

```
FOR counter = 500 TO 1000
  PULSOUT 14, counter
  PAUSE 20
NEXT
```

Il ciclo **FOR…NEXT** fa muovere la crociera del Servomotore a partire dalla posizione ore 2 ruotando lentamente fino a raggiungere la posizione ore 10. Siccome **counter** è l'indice del ciclo **FOR…NEXT,** esso aumenta di uno ad ogni iterazione. Il valore di counter viene anche usato nell'argomento ***Duration*** del comando **PULSOUT**, il ché implica che la ***duration*** di ciascun impulso diviene un poco più lunga ad ogni iterazione. Dal momento che ***duration*** cambia, altrettanto fa la posizione della crociera del Servomotore.

I cicli FOR…NEXT hanno un argomento opzionale **STEP**. L'argomento **STEP** può essere usato per far ruotare più velocemente il Servomotore. Per esempio, potete usare l'argomento **STEP** per aggiungere 8 a **counter** ad ogni iterazione (invece di 1) modificando la dichiarazione **FOR** nel seguente modo:

```
FOR counter = 500 TO 1000 STEP 8
```

Potete anche far girare il Servomotore nella direzione opposta contando all'indietro invece che in avanti. In PBASIC, i cicli **FOR…NEXT** contano anche all'indietro se l'argomento ***StartValue*** è più grande dell'argomento ***EndValue***. Segue un esempio di come far contare un ciclo **FOR…NEXT** da 1000 a 500:

```
FOR counter = 1000 TO 500
```

Potete combinare il conteggio all'indietro con un argomento **STEP** per far ruotare il Servomotore più rapidamente nella direzione oraria nel seguente modo:

```
FOR counter = 1000 TO 500 STEP 20
```

Il trucco per far ruotare il Servomotore a differenti velocità è di usare questi cicli **FOR…NEXT** per contare in avanti ed all'indietro con diverse grandezze di **STEP**. Il prossimo programma esempio usa queste tecniche per far ruotare la crociera del Servomotore avanti ed indietro a diverse velocità.

Programma di Esempio: ServomotoreVelocities.bs2

√ Caricate ed eseguite il programma.
√ Appena il programma gira, osservate i cambiamenti del valore di **counter** nel Terminale di Debug.

√ Osservate, inoltre il diverso comportamento del Servomotore nei due diversi cicli **FOR…NEXT**. Ambedue le direzioni del Servomotore ed i cambiamenti di velocità.

```
' Che cosa è un Microcontrollore- ServomotoreVelocities.bs2
' Fa Ruotare il Servomotore in senso antiorario lentamente, quindi in senso
' orario rapidamente.

' {$STAMP BS2}
' {$PBASIC 2.5}

counter        VAR     Word

DO

DEBUG "Incrementa di 8 la larghezza dell'impulso", CR

  FOR counter = 500 TO 1000 STEP 8
    PULSOUT 14, counter
    PAUSE 7
    DEBUG DEC5 counter, CR, CRSRUP
  NEXT

  DEBUG CR, "Decrementa di 20 la larghezza dell'impulso", CR

  FOR counter = 1000 TO 500 STEP 20
    PULSOUT 14, counter
    PAUSE 7
    DEBUG DEC5 counter, CR, CRSRUP
  NEXT

  DEBUG CR, "Ripetizione", CR

LOOP
```


Come funziona ServomotoreVelocities.bs2

Il primo ciclo **FOR…NEXT** conta in avanti da 500 a 1000 in passi di 8. Dal momento che la variabile **counter** viene usata come argomento **Duration** del comando **Pulsout**, la crociera del Servomotore ruota in senso antiorario con passi che sono il quadruplo del passo più piccolo possibile.

```
FOR counter = 500 TO 1000 STEP 8
  PULSOUT 14, counter
  PAUSE 7
  DEBUG DEC5 counter, CR, CRSRUP
NEXT
```

Perché PAUSE 7 invece di PAUSE 20? Il comando **DEBUG DEC5 counter, CR, CRSRUP** impiega circa 8 ms per essere eseguito. Questo significa che **PAUSE 12** manterrà il ritardo di 20 ms tra gli impulsi. Qualche esperimento prova e misura, ha mostrato che **PAUSE 7** ha dato al Servomotore il movimento più fluido. Il vostro Servomotore può essere leggermente diverso.

Ulteriori formattatori DEBUG e caratteri di controllo per il comando **DEBUG** sono disponibili, che visualizzano il valore della variabile **counter**. Questo valore viene visualizzato usando il formato a 5 cifre decimali (**DEC5**). Dopo la visualizzazione del valore, c'è un ritorno a capo (**CR**). Dopo il ritorno a capo, il formattatore **CRSRUP** (cursor up) manda il cursore indietro alla riga precedente. Ad ogni iterazione questo fa visualizzare il nuovo valore di counter al posto del vecchio valore.

Il secondo ciclo **FOR…NEXT** conta all'indietro da 1000 a 500 in passi di 20. In questo esempio la variabile **counter** viene anche usata come argomento del comando **PULSOUT, così la crociera del** Servomotore ruota in senso orario.

```
FOR counter = 1000 TO 500 STEP 20
  PULSOUT 14, counter
  PAUSE 7
  DEBUG DEC5 counter, CR, CRSRUP
NEXT
```

Il Vostro Turno – Regolare le velocità

√ Provate valori di **STEP** differenti per far girare il Servomotore a diverse velocità.
√ Rilanciate il programma dopo ciascuna modifica.
√ Osservate l'effetto prodotto da ciascun nuovo valore di **STEP** su quanto veloce gira la crociera del Servomotore.
√ Sperimentate con diversi valori ***Duration*** del comando **PAUSE** (tra 3 e 12) per trovare il valore che fa girare il Servomotore nella maniera più fluida.

ESERCIZIO #4: SERVOMOTORE CONTROLLATO DA UN TASTO

In questo Capitolo, avete scritto programmi che fanno muovere il Servomotore secondo una serie di movimenti predefiniti, ed avete controllato il Servomotore usando il Terminale di Debug. Ma potete anche programmare il BASIC Stamp per controllare il Servomotore con un tasto. In questo Esercizio potrete:

- Costruire un circuito per controllare il Servomotore con un tasto.

- Programmare il BASIC Stamp per controllare il Servomotore con un tasto.

Quando lo avrete fatto, sarete in grado di premere un tasto che farà in modo che il BASIC Stamp faccia ruotare il Servomotore in una direzione, ed un altro tasto per far ruotare il Servomotore nell'altra direzione. Quando nessun tasto viene premuto, il Servomotore manterrà la posizione qualunque essa sia.

Componenti aggiuntivi per il Servomotore Controllato dai Tasti

Vengono usati gli stessi componenti degli esercizi precedenti in questo Capitolo. Per i circuiti dei tasti vi dovrete recuperare i seguenti componenti:

(2) Tasti – normalmente aperti
(2) Resistenze – 10 kΩ (marrone-nero-arancio)
(2) Resistenze – 220 Ω (rosso-rosso-marrone)
(3) Ponticelli

Aggiunta dei circuiti di controllo dei tasti

La Figura 4-20 mostra i circuiti dei tasti che userete per controllare il Servomotore.

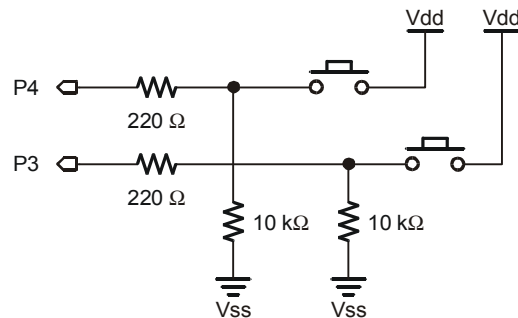



Figura 4-20
Circuiti dei Tasti per il controllo del Servomotore.

√ Aggiungete questo circuito al circuito del Servomotore + LED che avete usato fino ad ora. Quando lo avete finito il vostro circuito deve assomigliare:

- Alla Figura 4-21 se state usando la Board of Education Rev C
- Alla Figura 4-22 se state usando la HomeWork Board
- Alla Figura 4-23 se state usando la Board of Education Rev B

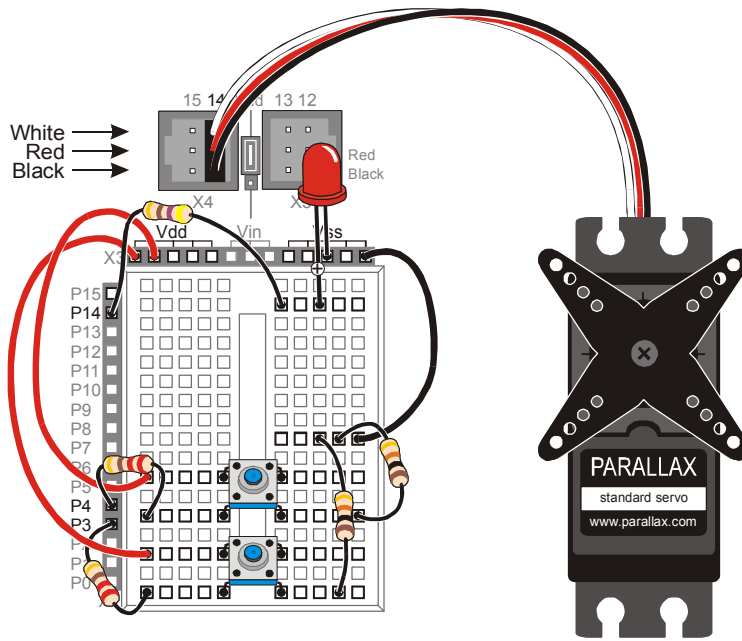



Figura 4-21
Board of Education Rev C circuito del Servomotore con aggiunto il circuito dei tasti.

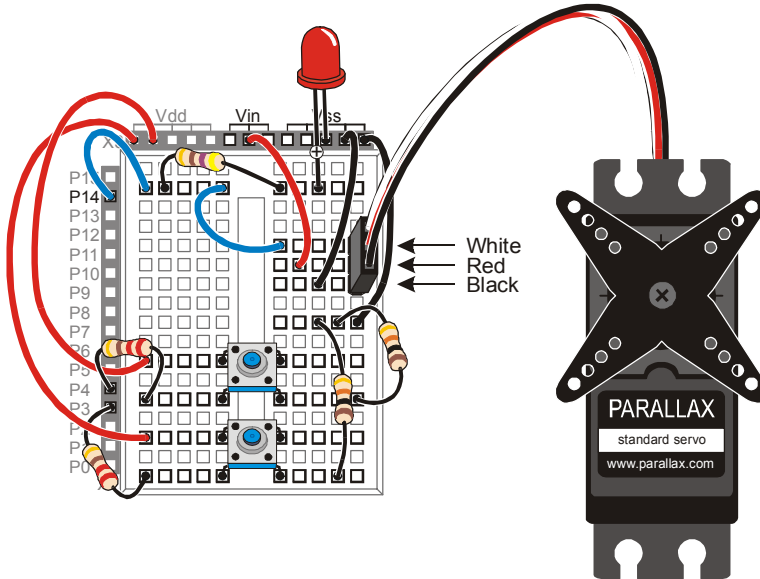



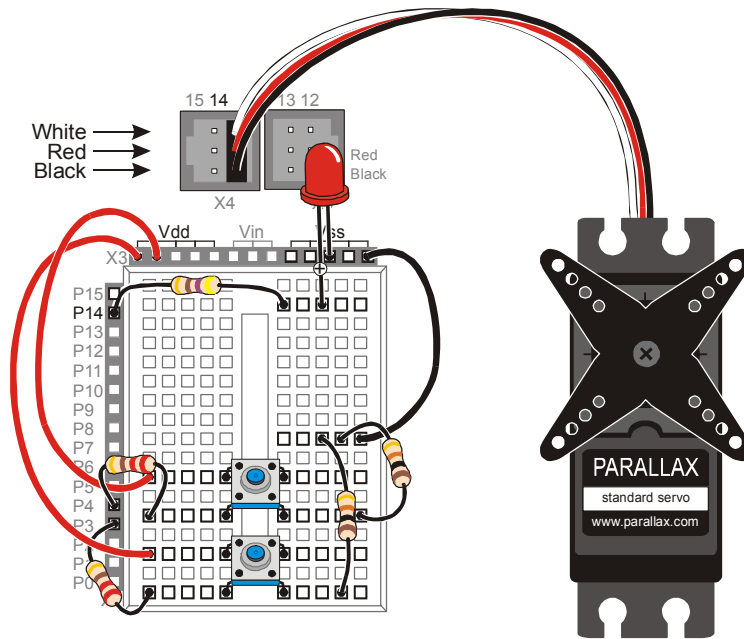
Figura 4-22
HomeWork Board circuito del Servomotore con aggiunto il circuito dei tasti.

4

Figura 4-23
Board of Education Rev B circuito del Servomotore con aggiunto il circuito dei tasti.

√ Collaudare il tasto collegato a P3 usando la versione originale di ReadTastoState.bs2. La sezione che contiene questo programma e le istruzioni per usarlo cominciano a pagina 81.
√ Modificate il programma in modo che legga P4.
√ Avviate il programma modificato per controllare il tasto collegato a P4.

Programmare il controllo del Tasto per il Servomotore

Il controllo del Tasto per il Servomotore non è molto diverso dal controllo del LED con un tasto. I blocchi di codice **IF…THEN** vengono usati per controllare lo stato dei tasti ed aggiungere o sottrarre da una variabile chiamata **duration**. Questa variabile viene usata nell'argomento ***Duration*** del comando **PULSOUT**. Se uno dei tasti viene premuto, il valore di **duration** aumenta. Se viene premuto l'altro tasto, il valore di **duration diminuisce**. Viene usato un ciclo annidato di istruzioni **IF…THEN** per decidere se la variabile **duration** è troppo grande (maggiore di 1000) o troppo piccola (minore di 500).

Programma di Esempio: ServomotoreControlWithTastos.bs2

Questo programma esempio fa ruotare la crociera del Servomotore in senso antiorario quando viene premuto il tasto collegato a P4. La crociera del Servomotore continuerà a girare fintanto che il tasto viene mantenuto premuto ed il valore di **`duration`** è minore di 1000. Quando invece viene premuto il tasto collegato a P3, la crociera del Servomotore ruota in senso antiorario. Il Servomotore viene inoltre limitato nel suo movimento orario perché alla variabile **`duration`** non è consentito andare al di sotto di 500. Il Terminale di Debug visualizza il valore di **`duration`** mentre il programma sta girando.

- √ Digitate il programma ServomotoreControlWithTastos.bs2 nell'Editor del BASIC Stamp ed avviatelo.
- √ Verificate che il Servomotore giri in senso antiorario quando premete e mantenete premuto il tasto collegato a P4.
- √ Verificate che appena il limite di **`duration`** > 1000 viene raggiunto o superato, il Servomotore smette di girare ulteriormente nella direzione antioraria.
- √ Verificate che il Servomotore giri in senso orario quando premete e mantenete premuto il tasto collegato a P3.
- √ Verificate che appena il limite di **`duration`** < 500 viene raggiunto o superato, il Servomotore smette di girare ulteriormente nella direzione oraria.
- √

```
' Che cosa è un Microcontrollore- ServomotoreControlWithTastos.bs2
' Premere e tenere premuto il tasto P4 per far ruotare il Servomotore in
senso.
' antiorario,
' Oppure premere e tenere premuto il tasto P3 per far ruotare il Servomotore
' in senso orario,

' {$STAMP BS2}
' {$PBASIC 2.5}

duration       VAR     Word

duration = 750

DO

  IF IN3 = 1 THEN
    IF duration > 500 THEN
      duration = duration - 25
    ENDIF
  ENDIF
```

```
  IF IN4 = 1 THEN
    IF duration < 1000 THEN
      duration = duration + 25
    ENDIF
  ENDIF

  PULSOUT 14, duration
  PAUSE 10

  DEBUG HOME, DEC4 duration, " = duration"

LOOP
```


Il Vostro Turno – Fine-Corsa via Software

I Servomotori hanno dei fine-corsa meccanici che gli impediscono di girare troppo. Se tentate di inviare un comando come **`PULSOUT 14, 2000`**, il Servomotore non girerà fino alla posizione che corrisponde all'argomento ***`Duration`*** di valore 2000. Questo perché i Servomotori hanno dei fine-corsa meccanici che limitano l'ampiezza del movimento. Facendo girare delicatamente la crociera, potete sentire quando il meccanismo interno del Servomotore arriva a questo fine-corsa meccanico. Potete modificare il programma esempio di questo Esercizio cosi che il BASIC Stamp limiti il movimento del Servomotore ad una ampiezza minore dei limiti imposti dai fine- corsa meccanici.

- √ Salvate ServomotoreControlWithTastos.bs2 con un nuovo nome.
- √ Regolate i limiti software imposti sul movimento del Servomotore in modo che siano 650 e 850 invece di 500 e 1000.
- √ Regolate la velocità impostata via software in maniera che la variabile **`duration`** venga incrementata o decrementata di 10 invece di 25.
- √ Decidete quale differenza vi aspettate di vedere nel comportamento del Servomotore quando premete un tasto.
- √ Lanciate il programma e paragonate i risultati ottenuti con la stima da voi fatta.

SOMMARIO

Questo Capitolo ha presentato il movimento microcontrollato tramite l'uso di un Servomotore. Un Servomotore è un dispositivo che si muove e si ferma in una particolare posizione in seguito ai segnali elettronici che riceve. Questi segnali hanno la forma di impulsi che durano un tempo variabile tra 1 e 2 ms, e devono essere inviati ogni 20 ms per far sì che il Servomotore rimanga nella posizione.

Un programmatore può usare il comando **`PULSOUT`** per far inviare questi segnali dal BASIC Stamp. Dal momento che per far mantenere la posizione questi segnali devono essere inviati ogni 20 ms, i comandi **`PULSOUT`** e **`PAUSE`** sono di regola inseriti in una qualche forma di ciclo iterativo. Le variabili possono essere usate per memorizzare il valore dell'argomento ***`Duration`*** dei comandi **`PULSOUT`**. Questo farà ruotare a passi la crociera del Servomotore.

In questo Capitolo, sono stati presentati una varietà di modi per inserire i valori nelle variabili. La variabile può ricevere il valore dal Terminale di Debug usando il comando **`DEBUGIN`**. Il valore della variabile può essere una sequenza di valori se la stessa variabile viene usata come indice in un ciclo **`FOR…NEXT`**. Questa tecnica può essere usata per far fare al Servomotore dei movimenti consecutivi. Le istruzioni **`IF…THEN`** possono essere usate per rilevare lo stato dei tasti ed aggiungere o sottrarre valori nell'argomento ***`Duration`*** del comando **`PULSOUT`** in base alla pressione o meno di un determinato tasto. Questo permette il controllo sia della posizione che della velocità in funzione del modo in cui il programma è strutturato e di come sono azionati i tasti.

Domande

1. A quali applicazioni o strumenti che contengano un microcontrollore che controlla il movimento vi siete affidati nell'ultima settimana?
2. Quali sono le quattro parti esterne di un Servomotore? A che cosa servono?
3. È necessario un circuito LED per far funzionare un Servomotore?
4. Quali comandi controllano il tempo basso del segnale inviato al Servomotore? E quali comandi controllano il tempo alto?
5. Quali elementi di programmazione potete usare per controllare per quanto tempo un servomotore mantiene una particolare posizione?
6. Quali comandi effettivamente controllano la posizione del Servomotore? Qual è il nome dell'argomento nel comando che, se cambiato, farà cambiare la posizione del Servomotore?

7. Quando un Servomotore è sotto il controllo del BASIC Stamp, che cosa vi indica la luminosità del LED circa il segnale che sta inviando al Servomotore?
8. Come usate il Terminale di Debug per inviare al BASIC Stamp? Quale comando di programmazione viene usato per far ricevere al BASIC Stamp messaggi dal Terminale di Debug?
9. Se l'utilizzatore sta inviando messaggi che dicono al Servomotore verso quale posizione girare, quale genere di blocco di codice vi aiuta ad assicurarvi che lui/lei non inserisca un numero che sia troppo alto o troppo basso?
10. Qual è il nome dell'argomento usato per far aggiungere o sottrarre ad un ciclo **FOR…NEXT** un valore maggiore di 1 ad ogni iterazione? Come fate a far contare all'indietro un ciclo **FOR…NEXT**?
11. Che cosa fa il Servomotore se usate l'indice di un ciclo **FOR…NEXT** come argomento ***Duration*** del comando **PULSOUT**?
12. Quale tipo di blocco di codice scrivereste per limitare l'ampiezza del movimento del Servomotore?

Esercizi

1. Il comando **PULSOUT 14, 750** invia un impulso di 1.5 ms sul pin I/O P14. Calcolate quanto durerà l'impulso se l'argomento ***Duration*** da 750 viene cambiato a 600. Ripetete questo calcolo per i seguenti valori: (a) 650, (b) 50000, (c) 1, (d) 2, (e) 2000.
2. Scrivete il comando necessario per inviare un impulso di 5.25 ms sul pin I/O P15.
3. Scrivete un ciclo che invia un impulso di 10.125 ms sul pin I/O P15 ogni 50 ms: (a) indefinitamente, (b) per 10 volte, (c) indefinitamente, fino a che il tasto collegato a P3 invia un segnale basso, (d) per dieci secondi dopo che venga rilevata la pressione di un tasto collegato a P4.
4. Scrivete un blocco di codice che cambi gradualmente il valore del comando **PULSOUT** che controlla un Servomotore da una durata di 700 fino a 800, quindi indietro fino a 700, in incrementi di (a) 1, (b) 2, (c) 3, (d) 4.
5. Aggiungete alla vostra risposta dell'esercizio 4 (d) un ciclo di **FOR…NEXT** annidato in modo che invii dieci impulsi prima di incrementare di 4 l'argomento ***Duration*** del comando **PULSOUT**.

Progetti

1. Modificate ServomotoreControlWithDebug.bs2 che controlli un tasto di blocco. Quando il tasto di blocco (il tasto collegato a P3) viene premuto dall'utente, il

Terminale di Debug non deve accettare ulteriori comandi. Dovrebbe visualizzare il messaggio: "Premere il tasto Avvio per avviare la macchina". Quando il tasto Avvio (il tasto collegato a P4) viene premuto, il programma funzionerà normalmente, come fa quando avviate il programma esempio. Se viene tolta l'alimentazione e ricollegata, il programma si deve comportare come se fosse stato premuto il tasto di blocco. OPZIONE: Aggiungere un circuito a LED bicolore collegato tra P12 e P13. Il LED bicolore, deve essere rosso dopo la pressione del tasto di blocco e verde dopo la pressione del tasto di Avvio.

2. Progettare un prototipo per un tergicristallo controllato da un tasto. Il tergicristallo dovrebbe avere sei velocità. Fermo ed intermittente (ogni 10 secondi) sono due di queste velocità. Le altre quattro fanno andare il tergicristallo avanti ed indietro all'incirca alle seguenti velocità: una volta ogni tre secondi, una volta ogni due secondi, una volta al secondo, e due volte al secondo. L'utilizzatore, dovrebbe premere il tasto collegato a P4 per far andare il tergicristallo più velocemente, e lui/lei dovrebbe premere il tasto collegato a P3per far andare il tergicristallo più lentamente.

Ulteriori Approfondimenti

Il Servomotore, e l'uso di sensori per controllare un Servomotore, può essere approfondito più in dettaglio in una varietà di testi della serie Stamps in Class.

"*Advanced Robotics: with the Toddler*", Student Guide, Version 1.2, Parallax Inc., 2003

Advanced Robotics: with the Toddler usa i Servomotori per controllare il movimento delle gambe del robot Toddler della Parallax. Sebbene diamo per scontato il camminare, programmare una macchina per farla camminare, manovrare, rimanere bilanciata può essere un'ardua sfida. Questo robot camminante viene raccomandato per studenti avanzati, che abbiano già la padronanza dei concetti espressi in *Che cosa è un Microcontrollore?* ed anche *Robotics!* o *SumoBot*.

"*Robotics!*", Student Workbook, Version 1.5, Parallax Inc., 2000

Robotics! Fa uso degli stessi principi di controllo del Servomotore che avete appena appreso con una differenza; I Servomotori possono anche essere modificati per essere usati come motori di spostamento dei robot. Usando il BASIC Stamp, la Board of Education, ed il kit Robotics!, questo testo inizia con i basilari della navigazione del robot, quindi vi guida attraverso la navigazione

mediante sensori. Introduce inoltre alcuni argomenti approfonditi come la soluzione di problemi di intelligenza artificiale e la navigazione tramite l'uso di sistemi di controllo elementari.

"*SumoBot*", Student Workbook, Version 1.1, Parallax Inc., 2002

Il Sumo Robot è una gara molto appassionante divertente per gli spettatori ed i partecipanti. *SumoBot* è un percorso guidato attraverso la costruzione, il controllo e la competizione con il vostro robot autonomo della classe Mini-Sumo. Questo libro di testo offre una presentazione stringata del testo *Robotica!,* Dei materiali impiegati con lo scopo di vincere una gara di lotta sumo robotica.

Capitolo #5: Misurare la rotazione

REGOLARE MANOPOLE E MONITORARE MACCHINE

Molti appartamenti hanno delle manopole per controllare l'illuminazione di una stanza. Ruotate la manopola in una direzione, e la luce aumenta; ruotate la manopola nell'altra direzione, e la luce si attenua. I trenini giocattolo usano le manopole per controllare la velocità del motore e la direzione. Molte macchine hanno manopole o leve usate per regolare con precisione la posizione di lame di taglio o di superfici guida.

Le manopole possono anche essere trovate negli apparati audio, dove sono usate per regolare come suonano la voce od il suono. La Figura 5-1 mostra un semplice esempio di manopola che viene ruotata per regolare il volume degli altoparlanti. Girando la manopola, un circuito interno alla cassa cambia, ed il volume della musica che gli altoparlanti suonano cambia. Circuiti simili possono anche essere trovati nei joystick, e perfino dentro i Servomotori usati nel Capitolo #4: Controllare il Movimento.

Figura 5-1
Regolatore del Volume in un Altoparlante

LA RESISTENZA VARIABILE DIETRO LA MANOPOLA – UN POTENZIOMETRO

Il dispositivo dietro alle manopole di un sistema audio, i joystick ed i Servomotori viene chiamato potenziometro, spesso abbreviato in "pot". La Figura 5-2 mostra la fotografia di alcuni potenziometri comuni. Notate che tutti quanti hanno tre piedini.

Figura 5-2
Esempio di alcuni Potenziometri

La Figura 5-3 mostra il simbolo elettrico ed il disegno pratico di un potenziometro che userete in questo Capitolo. I Terminali A e B sono collegati ad un elemento resistivo da 10 kΩ. Il Terminale W viene chiamato spazzola, ed è collegato ad un filo che tocca l'elemento resistivo in una posizione variabile tra le due estremità.

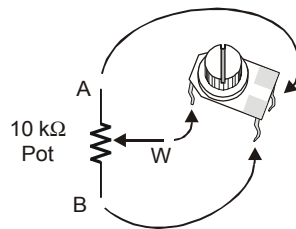



Figura 5-3
Simbolo elettrico e disegno pratico del potenziometro

La Figura 5-4 mostra come funziona la spazzola di un potenziometro. Quando muovete la manopola collegata ad un potenziometro, la spazzola collega un punto diverso dell'elemento resistivo. Quando muovete la manopola in senso orario, la spazzola si avvicina al terminale A, e quando girate la manopola in senso antiorario, la spazzola va verso il terminale B.

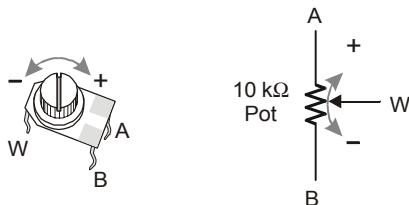



Figura 5-4
Spostamento della spazzola del Potenziometro

ESERCIZIO #1: COSTRUZIONE E COLLAUDO DEL CIRCUITO CON IL POTENZIOMETRO

Mettere resistenze di valore diverso in un circuito LED fa scorrere nel circuito differenti quantità di corrente. Resistenze di grande valore fanno scorrere poca corrente nel circuito, ed il LED si accende fiocamente. Resistenze di piccolo valore fanno scorrere nel circuito molta corrente, ed il LED diventa più luminoso. Collegando i terminali W ed A del potenziometro, in serie al circuito del LED, lo potete usare per regolare la resistenza del circuito. Questo a sua volta regola la luminosità del LED. In questo Esercizio, userete il potenziometro come resistenza variabile per cambiare la luminosità del LED.

5

Componenti del circuito con il potenziometro

(1) Potenziometro – 10 kΩ
(1) Resistenza – 220 Ω (rosso-rosso-marrone)
(1) LED – qualsiasi colore
(1) Ponticello

Costruzione del circuito di prova del Potenziometro

La Figura 5-5 mostra un circuito che può essere usato per regolare la luminosità del LED con un Potenziometro.

√ Costruite il circuito mostrato in Figura 5-5.

Consiglio: Usate una pinzetta a becchi lunghi per raddrizzare le eventuali pieghe dei terminali del Potenziometro prima di inserirlo nella scheda prototipi. Quando i terminali del Potenziometro sono diritti, fanno un contatto migliore con gli zoccoli della scheda prototipi.

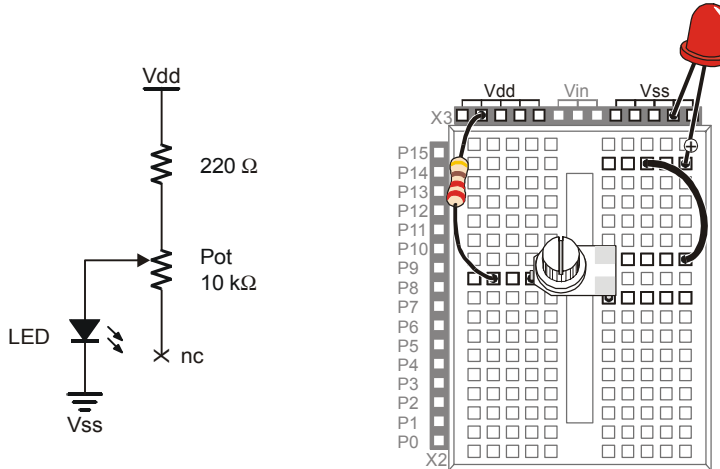



Figura 5-5
Circuito di prova
Potenziometro-LED

Collaudo del circuito del Potenziometro

√ Girare il Potenziometro in senso orario fino a che raggiunge il fermo meccanico mostrato in Figura 5-6.

Maneggiare con cura: Se il vostro Potenziometro non gira come mostrato, non lo forzate. Semplicemente giratelo fino al raggiungimento del fermo meccanico; altrimenti lo potreste rompere.

√ Gradualmente ruotate il Potenziometro in senso antiorario fino alla posizione mostrata in Figura 5-6 (b), (c), (d), (e), ed (f) notando come cambia la luminosità del LED a ciascuna posizione.

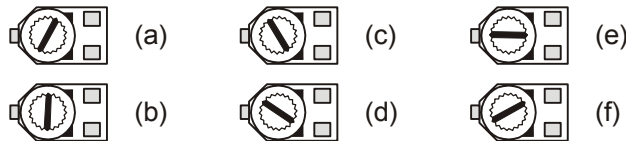



Figura 5-6
Alberino del Potenziometro

Le posizioni da (a) ad (f) mostrano la spazzola del Potenziometro messo in posizioni diverse.

Come funziona il circuito del Potenziometro

La resistenza totale nel vostro circuito di prova è di 220 Ω più la resistenza tra i terminali A e W del Potenziometro. Questo può essere un valore qualsiasi tra 0 e 10 kΩ. Quando girate la manopola del Potenziometro, la resistenza tra i terminali A e W cambia. Questo a sua volta cambia la corrente che scorre nel circuito del LED.

ESERCIZIO #2: MISURARE LA RESISTENZA MISURANDO IL TEMPO

5

In questo Esercizio viene inserito un nuovo componente chiamato condensatore. Un condensatore si comporta come una pila che però mantiene la sua carica per un breve periodo di tempo. Questo Esercizio introduce anche RC-time, che è l'abbreviazione per Resistenza-Condensatore Tempo. RC-time è la misura di quanto tempo impiega un condensatore a perdere una parte della sua carica mentre alimenta una Resistenza. Misurando il tempo che il condensatore impiega per scaricarsi con resistenze di diverso valore, diverrete più esperti nell'uso di RC-time. In questo Esercizio, programmerete il BASIC Stamp per caricare un condensatore e quindi misurare il tempo impiegato per scaricarsi attraverso una Resistenza.

Conosciamo il Condensatore

La Figura 5-7 mostra il simbolo elettrico ed il disegno pratico per il tipo di condensatore usato in questo Esercizio. Il valore di Capacità si misura in microfarad (μF), e la misura è normalmente stampata sui condensatori.

Questo condensatore ha un terminale positivo (+) ed uno negativo (-). Il terminale negativo è il terminale che esce dal contenitore dal lato vicino alla striscia con un segno meno (–). Assicuratevi sempre di collegare questi terminali come mostrato negli schemi elettrici. Collegare un condensatore in maniera errata, può danneggiarlo. In alcuni circuiti, collegare questo tipo di condensatori in maniera errata ed alimentandoli, può causare la loro rottura o perfino la loro esplosione.

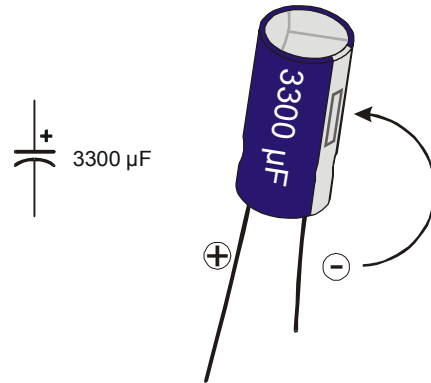



Figura 5-7
Condensatore da 3300 µF Schema elettrico e disegno pratico

Ponete molta attenzione ai terminali ed a dove devono essere collegati i terminali positivo e negativo.

Componenti per il circuito Resistenza e Tempo

(1) Condensatore – 3300 µF
(1) Condensatore – 1000 µF
(1) Resistenza – 220 Ω (rosso-rosso-marrone)
(1) Resistenza – 470 Ω (giallo-viola-marrone)
(1) Resistenza – 1 kΩ (marrone-nero-rosso)
(1) Resistenza – 2 kΩ (rosso-nero-rosso)
(1) Resistenza – 10 kΩ (marrone-nero-arancio)

Equipaggiamento Raccomandato: Occhiali normali oppure Occhiali di protezione.

Costruzione e Collaudo del circuito Resistenza Condensatore Tempo (RC-Time).

La Figura 5-8 mostra lo schema elettrico del circuito e la Figura 5-9 mostra lo schema di cablaggio per questo Esercizio. Farete misure di tempo usando Resistenze di valore diverso al posto della Resistenza etichettata R_i.

SICUREZZA

Rispettate sempre la polarità quando collegate il condensatore da 3300 μF. Ricordate, il terminale negativo è il filo che esce dalla parte più vicina alla striscia che riporta il segno meno (–). Usate la Figura 5-7 per identificare i terminali più (+) e meno (-).

Il vostro condensatore da 3300 μF funzionerà benissimo in questo esperimento, se vi assicurerete che i suoi terminali positivo (+) e negativo (-) siano collegati ESATTAMENTE come mostrato in Figura 5-8 ed in Figura 5-9.

Non Invertire MAI la Polarità sul condensatore da 3300 μF o su qualsiasi altro condensatore elettrolitico. La tensione al terminale (+) deve sempre essere più alta della tensione al suo terminale (-). Vss è la tensione più bassa (0 V) sulla Board of Education e sulla BASIC Stamp HomeWork Board. Collegando il terminale negativo del condensatore a Vss, sarete sicuri che la polarità ai capi dei terminali del condensatore sarà sempre corretta.

Durante questo Esercizio indossate un paio di occhiali o una maschera di sicurezza.

Spengete sempre prima di costruire o modificare questo circuito.

Quando alimentate il circuito allontanate le mani e la faccia da questo condensatore.

5

√ Con l'alimentazione scollegata, costruite il circuito come mostrato iniziando dalla Resistenza da 470 Ω al posto di quella etichettata R_i.

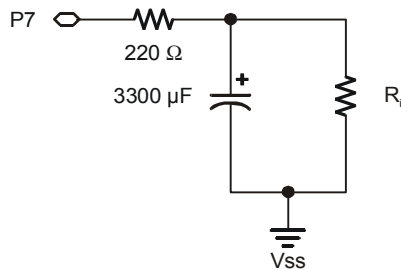



Figura 5-8
Schema elettrico per vedere il decadimento della tensione rc-time.

Saranno usate quattro diverse Resistenze al posto di R_i mostrata nello schema. Per primo lo schema sarà costruito e collaudato con R_i = 470 Ω, quindi R_i = 1 kΩ, etc.

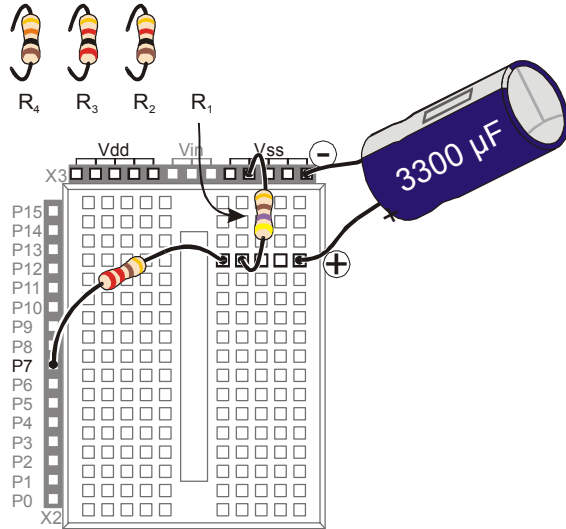



Figura 5-9
Schema pratico della Figura 5-8

Assicuratevi che il terminale negativo del condensatore sia connesso sulla vostra scheda, nello stesso modo mostrato in questa Figura, con il negativo collegato a Vss.

Misurare il tempo del circuito RC-Time con il BASIC Stamp

Sebbene possa essere usato un cronometro per misurare in quanto tempo la carica del condensatore decade ad un certo livello, il BASIC Stamp può anche essere programmato per tenere sotto controllo il circuito e darvi una misura del tempo più affidabile.

Programma di Esempio: PolledRcTimer.bs2

√ Digitate e lanciate PolledRcTimer.bs2.
√ Osservate come il BASIC Stamp carica il condensatore e quindi misura il tempo di scarica.
√ Registrate il tempo misurato (il tempo di scarica del condensatore) nella fila della resistenza da 470 Ω nella Tabella 5-1.
√ Togliete l'alimentazione dalla vostra Board of Education o BASIC Stamp HomeWork Board.
√ Rimuovete la resistenza da 470 Ω etichettata R_i nella Figura 5-8 e nella Figura 5-9 a pagina 147, e sostituitela con la Resistenza da 1 kΩ.
√ Ridate alimentazione.
√ Registrate la misura del tempo (per la Resistenza da 1 kΩ).
√ Ripetete questi passi per ciascun valore di Resistenza della Tabella 5-1.

Tabella 5-1: Resistenza ed RC-time per C = 3300 μF	
Resistenza (Ω)	**Tempo Misurato (s)**
470	
1 k	
2 k	
10 k	


```
' Che cosa è un Microcontrollore- PolledRcTimer.bs2
' Programma di misura del Tempo di Reazione modificato per la misura del
' tempo di decadimento della tensione RC-time.

' {$STAMP BS2}
' {$PBASIC 2.5}

timeCounter    VAR     Word
counter        VAR     Nib

DEBUG CLS

HIGH 7
DEBUG "Carica del Condensatore...", CR

FOR counter = 5 TO 0
  PAUSE 1000
  DEBUG DEC2 counter, CR, CRSRUP
NEXT

DEBUG CR, CR, "Misura del Tempo di Decadimento!", CR, CR
INPUT 7

DO

  PAUSE 100
  timeCounter = timeCounter + 1

  DEBUG ? IN7
  DEBUG DEC5 timeCounter, CR, CRSRUP, CRSRUP

LOOP UNTIL IN7 = 0

DEBUG CR, CR, CR, "Il Tempo di Decadimento RC era di ",
      DEC timeCounter, CR,
      "decimi di secondo.", CR, CR

END
```

Come funziona PolledRcTimer.bs2

Vengono dichiarate due variabili. La variabile **`timeCounter`** viene usata per tenere traccia del tempo impiegato dal condensatore per scaricarsi attraverso R_i. La variabile **`counter`** viene usata per contare mentre il condensatore si sta caricando.

```
timeCounter    VAR     Word
counter        VAR     Nib
```

Il comando **`DEBUG CLS`** pulisce lo schermo del Terminale di Debug in modo che non venga riempito dalle misure in successione. **`HIGH 7`** imposta P7 alto e comincia a caricare il condensatore, quindi viene visualizzato il messaggio **“**`Carica del Condensatore…`**”**. Dopo di che, un ciclo **`FOR…NEXT`** conta all’indietro mentre il condensatore si carica. Man mano che il condensatore si carica, la tensione ai capi dei terminali aumenta fino ad un livello posto tra 2.5 e 4.9 V (dipende dal valore di R_i).

```
DEBUG CLS

HIGH 7
DEBUG " Carica del Condensatore...", CR

FOR counter = 5 TO 0
  PAUSE 1000
  DEBUG DEC2 counter, CR, CRSRUP
NEXT
```

Un messaggio annuncia l’inizio della misura del tempo di scarica.

```
DEBUG CR, CR, " Misura del Tempo di Decadimento!", CR, CR
```

Per far sì che il condensatore si possa scaricare attraverso la Resistenza R_i, il pin I/O viene cambiato da **`HIGH`** in **`INPUT`**. Come ingresso, il pin I/O, non ha effetto sul circuito, ma può rilevare dei segnali alti o bassi. Dal momento che il pin I/O rilascia il circuito, il condensatore si scarica fornendo corrente alla Resistenza. Man mano che il condensatore si scarica, la tensione ai suoi capi diventa sempre più bassa (decade).

```
INPUT 7
```

Precedentemente nel Capitolo del tasto, avete usato il BASIC Stamp per rilevare un segnale alto o basso usando le variabili **`IN3`** ed **`IN4`**. In quel momento, un segnale alto era considerato Vdd, ed un segnale basso era considerato Vss. In realtà un segnale alto è una tensione sopra 1.4 V. Certamente, potrebbe arrivare fino a 5 V. Similmente, un segnale basso è qualsiasi tensione tra 1.4 V e 0 V. questo ciclo **`DO…LOOP`** controlla P7 ogni 100 ms

fino a che il valore di **IN7** cambia da 1 a 0, che indica che la tensione del condensatore è decaduta sotto 1.4 V.

```
DO

  PAUSE 100
  timeCounter = timeCounter + 1

  DEBUG ? IN7
  DEBUG DEC5 timeCounter, CR, CRSRUP, CRSRUP

LOOP UNTIL IN7 = 0
```

Il risultato viene quindi visualizzato ed il programma termina.

```
DEBUG CR, CR, CR, "Il tempo di decadimento RC era",

      DEC timeCounter, CR,
      "Decimi di secondo.", CR, CR

END
```


Il Vostro Turno – Un Circuito più Veloce

Usando un condensatore che abbia grosso modo 1/3 della capacità per mantenere la carica, il tempo misurato per ciascun valore di resistenza sarà ridotto ad 1/3. Nell' Esercizio #3, userete un condensatore che è 10,000 volte più piccolo, ed il BASIC Stamp farà la misura del tempo per voi usando il comando chiamato **RCTIME**.

- √ Spengete la vostra Board of Education o HomeWork Board
- √ Sostituite il condensatore da 3300 μF con un condensatore da 1000 μF.
- √ **Controllate che la polarità del vostro condensatore sia corretta.** Il terminale negativo deve essere collegato a Vss.
- √ Ridate alimentazione.
- √ Ripetete i passi nella sezione del Programma di Esempio: PolledRcTimer.bs2, e registrate le vostre misure di tempo nella Tabella 5-2.
- √ Paragonate le ultime misure di tempo con quelle prese precedentemente nella Tabella 5-1. Quanto si avvicinano ad 1/3 del valore delle misure prese con il condensatore da 3300 μF?

Tabella 5-2: Resistenza ed RC-time per C = 1000 µF	
Resistenza (Ω)	**Tempo Misurato (s)**
470	
1 k	
2 k	
10 k	

ESERCIZIO #3: LETTURA DELLA POSIZIONE DEL POTENZIOMETRO CON IL BASIC STAMP

Nell'Esercizio #1, è stato usato un Potenziometro come Resistenza variabile. La resistenza nel circuito variava in funzione della posizione della manopola del Potenziometro. Nell'Esercizio #2, un circuito RC-time è stato usato per misurare la differenza di resistenza. In questo Esercizio, costruirete un circuito RC-time per leggere il Potenziometro, ed userete il BASIC Stamp per fare misure di tempo. Il condensatore che userete sarà molto piccolo, e la misura del tempo durerà solamente pochi millisecondi. Sebbene le misure vengano fatte in un tempo molto breve, il BASIC Stamp vi darà un'eccellente misura della resistenza esistente tra i terminali A e W del Potenziometro.

Componenti per la lettura di RC-Time con il BASIC Stamp

(1) Potenziometro – 10 kΩ
(1) Resistenza – 220 Ω (rosso-rosso-marrone)
(2) Ponticelli
(1) Condensatore – 0.1 µF mostrato in Figura 5-10
(1) Condensatore – 0.01 µF, anche lui mostrato in Figura 5-10
(2) Ponticelli

Questi condensatori non hanno terminali + e –. Potete collegare in maniera sicura questi condensatori ad un circuito senza preoccuparvi della polarità.

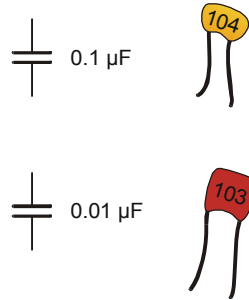



Figura 5-10
Condensatori Ceramici

Il condensatore da 0.1 µF (sopra) ed il condensatore da 0.01 µF (sotto) sono ambedue non polarizzati. Con questi due componenti non vi dovete preoccupare del terminale positivo o negativo.

5

Costruire il circuito RC Time per il BASIC Stamp

La Figura 5-11 mostra lo schema elettrico per il circuito RC-time veloce, e la Figura 5-12 mostra lo schema pratico. Questo è il circuito che userete per rilevare la posizione del Potenziometro con l'aiuto del BASIC Stamp e di un programma PBASIC.

√ Costruite il circuito mostrato in Figura 5-11.

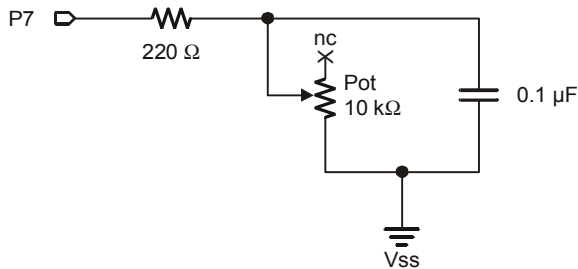



Figura 5-11
Circuito RCTIME per il BASIC Stamp con Potenziometro

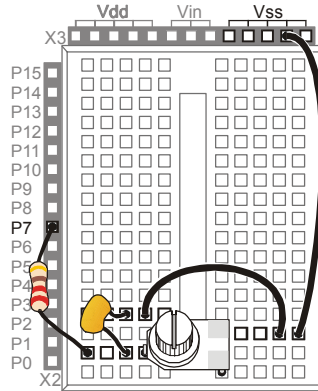



Figura 5-12
Schema di cablaggio per la Figura 5-11

Programmazione delle misure RC-Time

Il programma BASIC Stamp per la misura della resistenza del Potenziometro farà essenzialmente la stessa cosa che avete fatto a mano nell'Esercizio #2. La pressione ed il mantenimento della pressione sono equivalenti ad un comando **HIGH** seguito da una **PAUSE**. Il comando **RCTIME** è il modo che il BASIC Stamp usa per lasciare andare il tasto e misurare il tempo fino a che la tensione si abbassi a sufficienza per passare la soglia di **IN7** (1.4 V).

Programma di Esempio: ReadPotWithRcTime.bs2

√ Digitate e lanciate ReadPotWithRcTime.bs2
√ Provate a ruotare la manopola del Potenziometro mentre controllate il valore della variabile **time** usando il Terminale di Debug.

```
' Che cosa è un Microcontrollore- ReadPotWithRcTime.bs2
' Legge il Potenziometro in un circuito RC-time usando il comando RCTIME.

' {$STAMP BS2}
' {$PBASIC 2.5}

time VAR Word

DO

  HIGH 7
  PAUSE 100
  RCTIME 7, 1, time
  DEBUG HOME, "tempo = ", DEC5 time
```

```
LOOP
```

Come funziona ReadPotWithRcTime.bs2

Di seguito ci sono i passi di pseudo codice che il programma compie per fare la misura del tempo RC-time.

- *Dichiarare la variabile **`time`** per memorizzare la misura del tempo.*
- *Blocco di Codice nel ciclo **`DO…LOOP`**:*
    - *Imposta il pin I/O P7 a **`HIGH`**.*
    - *Attendi per 100 ms (20 ms per assicurarsi che il condensatore si carichi ed ulteriori 80 ms per dar tempo al Terminale di Debug di visualizzare).*
    - *Eseguire il comando **`RCTIME`**.*
    - *Memorizzare la misura di tempo nella variabile **`time`**.*
    - *Visualizza il valore **`time`** nel Terminale di Debug.*

Prima di eseguire il comando **`RCTIME`**, il condensatore deve essere completamente caricato. Non appena viene eseguito il comando **`RCTIME`**, il BASIC Stamp cambia il pin I/O pin da uscita ad ingresso. Come ingresso, il pin I/O assomiglia al circuito dell'Esercizio #2 quando il tasto veniva rilasciato (circuito aperto). Il comando **`RCTIME`** è la versione veloce del sistema di rilevazione che è stata usata nell'Esercizio #2, e misura la quantità di tempo che impiega il condensatore a perdere la sua carica e decadere al di sotto della soglia di 1.4 V del pin I/O. Invece di contare con incrementi di 100 ms, il comando **`RCTIME`** conta con incrementi di 2 μs.

Il Vostro Turno – Cambiare il Tempo cambiando il Condensatore

√ Sostituire il condensatore da 0.1 μF con un condensatore da 0.01 μF.

√ Provare con le stesse posizioni del Potenziometro che avete provato nell'Esercizio principale e confrontate i valori visualizzati nel Terminale di Debug con i valori ottenuti con il condensatore da 0.1 μF. Le misure **`RCTIME`** sono un decimo del valore precedente?

√ Rimettete il condensatore da 0.1 μF.

√ Con il condensatore da 0.1 μF montato nel circuito ed il condensatore da 0.01 μF rimosso, annotate i valori più alto e più basso per le prove del prossimo Esercizio.

ESERCIZIO #4: CONTROLLARE UN SERVOMOTORE CON UN POTENZIOMETRO

I Potenziometri insieme con i Servomotori possono essere usati per una varietà di cose divertenti. Queste sono le basi per il modellismo, di aerei, auto e barche. Questo Esercizio mostra come il BASIC Stamp può essere usato per controllare la posizione di un Servomotore con un circuito Potenziometro.

Un esempio di aeromodello e del suo radiocomando sono mostrati in Figura 5-13. L'aeromodello ha dei servomotori per controllare la posizione dei flap e dell'acceleratore. Questi Servomotori sono controllati via radio con l'unità radio comando (RC) mostrato davanti all'aeromodello. Questa unità RC ha dei potenziometri sotto ad una coppia di joystick che sono usati per controllare i servomotore che a loro volta controllano i timoni di quota e di direzione dell'aereo.

Figura 5-13
Aeromodello e Radiocomando

Come l'unità RC controlla l'Aeromodello: I Potenziometri all'interno dei joystick sono monitorati da un circuito che converte la posizione del joystick in impulsi per il Servomotore. Questi impulsi di controllo sono quindi convertiti in segnali radio e trasmessi dal radiocomando a terra ad un ricevitore nell'Aeromodello. Questo ricevitore riconverte i segnali radio in impulsi di controllo che posizionano quindi la crociera dei Servomotori.

Componenti per il Servomotore Controllato dal Potenziometro

(1) Potenziometro – 10 kΩ

(1) Resistenza – 220 Ω (rosso-rosso-marrone)
(1) Condensatore – 0.1 μF
(1) Servomotore
(1) LED – qualsiasi colore
(2) Ponticelli

Agli utilizzatori della HomeWork Board servirà inoltre:

(1) Connettore a 3-pin maschio-maschio
(4) Ponticelli

Costruire i circuiti del potenziometro e del Servomotore

Questo Esercizio userà due circuiti che avete già costruito individualmente: il circuito del Potenziometro dell'Esercizio che avete appena finito ed il circuito con Servomotore del Capitolo precedente.

√ Lasciate il circuito Potenziometro RC-time dell'Esercizio #3 sulla vostra scheda prototipale. Se lo dovete riassemblare, usate la Figura 5-11 a pagina 153 e la Figura 5-12 a pagina 154. Assicuratevi di usare il condensatore da 0.1 μF, non il condensatore da 0.01 μF.
√ Aggiungete al progetto il circuito Servomotore dell' Esercizio #1 Capitolo #4. Ricordate che il vostro circuito del Servomotore sarà diverso secondo quale scheda state usando. Sono riportate sotto le pagine per la scheda a cui dovete riferirvi:

- Page 107 – Board of Education Rev C
- Page 113 – Board of Education Rev B
- Page 110 – BASIC Stamp HomeWork Board

Programmare il Controllo a Potenziometro del Servomotore

Vi serviranno il più grande ed il più piccolo dei valori della variabile **`time`** che avete registrato dal circuito RC-time usando il condensatore da 0.1 μF.

√ Se non avete ancora completato la sezione "Il Vostro Turno" dell'Esercizio precedente, fatelo ora.

Per questo prossimo esempio, di seguito ci sono i valori **`time`** che sono stati misurati dai tecnici della Parallax; i vostri valori saranno probabilmente leggermente diversi:

- Tutto in senso orario: 1
- Tutto in senso antiorario: 691

Come possono questi valori essere riferiti ai valori di 500 e 1000 che sono necessari per controllare il Servomotore con il comando **`PULSOUT`**? La risposta è usare la moltiplicazione e la somma. Per prima cosa, moltiplicate i valori di ingresso per portare la differenza tra i valori orario (minimo) ed antiorario (massimo) a 500 invece di quasi 700. Quindi, sommate al risultato un valore costante cosi che spazi da 500 a 1000 invece che da 1 a 500. In elettronica, queste operazioni sono chiamate scalatura ed offset.

Ecco le formule matematiche della moltiplicazione (scalatura):

$$time(maximum) = 691 \times \frac{500}{691} = 691 \times 0.724 = 500$$

$$time(minimum) = 1 \times \frac{500}{691} = 0.724$$

Dopo che i valori sono stati scalati, qui c'è l' addizione (offset).

$$time(maximum) = 500 + 500 = 1000$$

$$time(minimum) = 0.724 + 500 = 500$$

L'operatore **`*/`** che è stato discusso a pagina 99 è presente nel PBASIC per la scalatura con valori frazionali, come 0.724. di seguito sono riportate le operazioni per usare **`*/`** applicato a 0.724:

1. Mettete il valore o la variabile che volete moltiplicare per un valore frazionario prima dell'operatore */.

   ```
   time = time */
   ```

2. Prendete il valore frazionale che volete usare e moltiplicatelo per 256.

   $$new\ fractional\ value = 0.724 \times 256 = 185.344$$

3. Arrotondatelo alla prima cifra intera.

$$new\ fractional\ value = 185$$

4. Mettete il valore così ottenuto dopo l'operatore */.

```
time = time */ 185
```

Questo si occupa della scalatura, ora tutto quello che dovete fare è aggiungere il valore di offset di 500. Questo può essere fatto con un secondo comando che aggiunge 500 a **`time`**:


```
time = time */ 185
time = time + 500
```

Ora, **`time`** è pronto per essere inserito nell'argomento ***`Duration`*** del comando **`PULSOUT`**.

```
time = time */ 185        ' Scalatura per 0.724.
time = time + 500         ' Offset di 500.
PULSOUT 14, time          ' Invia l'impulso al Servomotore.
```

Programma di Esempio: ControlServomotoreWithPot.bs2

√ Digitate e lanciate questo programma, quindi ruotate la manopola del Potenziometro ed assicuratevi che i movimenti del Servomotore riflettano quelli del Potenziometro.

```
' Che cosa è un Microcontrollore- ControlServomotoreWithPot.bs2
' Legge il Potenziometro nel circuito RC-time usando il comando RCTIME.
' Scala time per 0.724 ed esegue l'offset a 500 per il Servomotore.

' {$STAMP BS2}
' {$PBASIC 2.5}

time           VAR     Word

DO
  HIGH 7
  PAUSE 10
  RCTIME 7, 1, time
  time = time */ 185                   ' Scalatura by 0.724 (X 256 for */).
  time = time + 500                    ' Offset di 500.
  PULSOUT 14, time                     ' Invia l'impulso al Servomotore.
LOOP
```

Il Vostro Turno – Adattare il Servomotore alla posizione del Potenziometro

I vostri Potenziometro e condensatore vi avranno probabilmente dato dei valori **`time`** in qualche maniera diversi da quelli di questo Esercizio. Sono i valori che avete raccolto nella sezione Il Vostro Turno dell'Esercizio precedente.

√ Usando i vostri valori massimo e minimo ripetete la matematica discussa nella sezione Programmare il Controllo a Potenziometro del Servomotore a pagina 157.

√ Sostituite i vostri valori **`scale`** ed **`offset`** in ControlServomotoreWithPot.bs2.

√ Aggiungete questa linea di codice tra i comandi **`PULSOUT`** e **`LOOP`** cosi che possiate vedere i vostri risultati.

```
DEBUG HOME, DEC5 time ' Visualizza i valori corretti di time.
```

√ Lanciate il programma modificato e controllate il vostro lavoro. A causa degli arrotondamenti, i limiti possono non essere esattamente 500 e 1000, ma ci dovrebbero essere vicini.

Usare le Costanti nei Programmi

Nei programmi più grandi, potreste finire con l'usare molte volte il comando **`PULSOUT`** ed il valore di scala (che era 185) e l'offset (che era 500). Per questi valori potete usare i nomi "alias" usando la direttiva **`CON`** come mostrato di seguito:

```
scaleFactor CON 185
offset CON 500
```

Questi nomi "alias" sono quasi sempre dichiarati nella parte iniziale del programma cosi che siano facili da trovare.

Ora, dovunque nel vostro programma vogliate usare uno di questi valori, potete usare al loro posto le parole **`offset`** o **`scaleFactor`**. Per esempio,

```
time = time */ scaleFactor                ' Scalatura di 0.724.
time = time + offset                      ' Offset di 500.
```

Potete anche applicare la stessa tecnica con i pin I/O. Per esempio, potete dichiarare una costante per il pin I/O P7.

```
rcPin    CON 7
```

Ci sono due casi nell'esempio precedente dove il numero 7 viene usato per riferirsi al pin I/O P7. Il primo può ora essere scritto:

```
HIGH rcPin
```

Ed il secondo può essere scritto:

```
RCTIME rcPin, 1, time
```

Se in seguito cambiate il circuito, tutto quello che dovrete fare è cambiare il valore nella dichiarazione delle costanti, ed ambedue i comandi **HIGH** e **RCTIME** saranno automaticamente aggiornati. Similmente, se dovete ricalibrare il vostro fattore di scala o l'offset, dovrete solamente cambiare le direttive **CON** all'inizio del programma.

5

Assegnare un alias è quello che fate quando date un nome ad una variabile, costante o pin I/O usando **VAR**, **CON**, o **PIN**.

Programma di Esempio: ControlServomotoreWithPotUsingConstants.bs2

Questo programma fa uso di alias al posto di quasi tutti i numeri.

√ Digitate e lanciate ControlServomotoreWithPotUsingConstants.bs2.
√ Osservate com'è la risposta del Servomotore al Potenziometro e verificate che sia la stessa dell'esempio precedente (ControlServomotoreWithPot.bs2).

```
' Che cosa è un Microcontrollore- ControlServomotoreWithPotUsingConstants.bs2
' Legge  la posizione del Potenziometro usando il comando RCTIME.
' Applica il fattore di scala e l'offset, quindi invia il valore al
' Servomotore.

' {$STAMP BS2}
' {$PBASIC 2.5}

scaleFactor    CON     185
offset         CON     500
rcPin          CON     7
delay          CON     10
ServomotorePin       CON     14

time           VAR     Word

DO
  HIGH rcPin
  PAUSE delay
  RCTIME rcPin, 1, time
```

```
  time = time */ scaleFactor        ' Fattore di Scala da scaleFactor.
  time = time + offset              ' Offset da offset.
  PULSOUT ServomotorePin, time      ' Invia l'impulso al Servomotore.
  DEBUG HOME, DEC5 time             ' Visualizza il valore corretto del tempo.
LOOP
```

Il Vostro Turno – Uso delle Costanti per la Calibrazione e l'aggiornamento facile

Come menzionato in precedenza, se cambiate il pin I/O usato dai comandi **HIGH** e **RCTIME**, potete semplicemente cambiare il valore di dichiarazione della costante **rcPin**.

- √ Salvate il programma esempio con un nuovo nome.
- √ Cambiate i valori di **scaleFactor** e **offset** con i valori precisi del vostro circuito RC che avete determinato nella sezione precedente de Il Vostro Turno.
- √ Lanciate il programma modificato e verificate che funzioni correttamente.
- √ Modificate il vostro circuito collegando il circuito RC-time dal pin I/O P7 al pin I/O P8.
- √ Modificate la dichiarazione **rcPin** nel modo seguente:

```
rcPin    CON 8
```

- √ Aggiungete questo comando, prima del ciclo **DO…LOOP** in modo che possiate vedere che la costante **rcPin** in realtà è un modo di indicare il numero otto:

```
DEBUG ? rcPin
```

- √ Rilanciate il programma e verificate che i comandi **HIGH** e **RCTIME** funzionino ancora correttamente sui diversi pin I/O solamente cambiando il valore della direttiva **rcPin CON**.

SOMMARIO

Questo Capitolo ha presentato il Potenziometro, un componente che si trova spesso dietro svariate manopole ed indicatori. Il Potenziometro ha un elemento resistivo normalmente collegato tra i suoi terminali estremi, e la spazzola che collega un punto qualsiasi dell'elemento resistivo. Il Potenziometro può essere usato come resistenza variabile se in un circuito vengono usati la spazzola ed uno dei due terminali estremi.

5

In questo Capitolo è stato anche presentato il condensatore. Un condensatore può essere usato per mantenere e rilasciare cariche elettriche. La quantità di cariche che un condensatore può mantenere dipende dal suo valore, che viene misurato in Farad, (F). La lettera greca μ è l'abbreviazione scientifica di micro, e significa un milionesimo. I condensatori usati negli esercizi di questo Capitolo hanno un valore che và da 0.01 a 3300 μF.

Una Resistenza ed un condensatore possono essere collegati insieme in un circuito che impiegherà un certo tempo a caricarsi ed a scaricarsi. Questo circuito viene sovente chiamato un circuito RC. La R e la C in RC stanno per Resistenza e Condensatore. Quando un valore (C negli esercizi di questo Capitolo) viene mantenuto costante, la variazione del tempo di carica e dovuta al valore di R. Quando il valore di R cambia, cambia anche il tempo che il circuito impiega a caricarsi ed a scaricarsi. Il tempo totale che impiega il circuito RC per scaricarsi può essere variato usando un condensatore di diversa capacità.

È stato rilevato il tempo di scarica di un condensatore in un circuito RC dove il valore di C era molto grande. Sono state usate resistenze di valore diverso per mostrare come cambia il tempo di scarica quando cambia il valore della Resistenza nel circuito. È stato usato il comando **RCTIME** per misurare un potenziometro (una Resistenza variabile) in un circuito RC con valori più piccoli per il condensatore. Sebbene questi condensatori abbiano un tempo di scarica che và da circa 2 a 1500 μs (milionesimi di secondo), il BASIC Stamp non ha problemi a tenere sotto controllo queste misure con il comando **RCTIME**. Il pin I/O deve essere posto **HIGH**, in modo che al condensatore nel circuito RC sia consentito caricarsi usando il comando **PAUSE** prima che il comando **RCTIME** possa essere usato.

La programmazione PBASIC può essere usata per misurare un sensore resistivo come un Potenziometro e per adattare il suo valore alle caratteristiche di un altro dispositivo, come

ad esempio un Servomotore. Questo implica il dover compiere operazioni matematiche sulla misura del tempo di scarica, che il comando **RCTIME** memorizza in una variabile. Questa variabile può essere modificata aggiungendogli una costante, il che torna comodo per controllare un Servomotore. Nella sezione Progetti, potrete anche trovarvi ad usare moltiplicazioni e divisioni. La direttiva **CON** può essere usata all'inizio di un programma per sostituire un numero con un nome. Come per i nomi delle variabili, nominare una costante viene detto creare un alias. Dopo che un alias è stato creato, il nome può essere usato in tutto il programma al posto del numero. Questo torna utile se dovete usare i soliti numeri in 2, 3 o anche 100 posti diversi del programma. Potete cambiare i numeri della direttiva **CON**, e tutte le 2, 3 o perfino 100 istanze di quel numero vengono automaticamente aggiornate la prossima volta che lanciate il programma.

Domande

1. Descrivete cinque esempi di dispositivi che usano manopole?
2. Quando ruotate la manopola di un sistema audio, quale componente state probabilmente regolando?
3. In un Potenziometro tipico, la resistenza ai capi dei due terminali estremi è regolabile?
4. Quando viene ruotato l'alberino di un Potenziometro, che cosa avviene a ciascuno dei tre terminali della resistenza? Quale relazione esiste tra le tre possibili coppie di terminali? Quale effetto ha su un circuito LED collegato ai terminali A e W del Potenziometro?
5. Quanto un condensatore è simile ad una batteria ricaricabile? E quanto è diverso?
6. Che cosa dovete fare con un circuito RC-time per avere l'indicazione del valore di una resistenza variabile?
7. Ammettiamo di avere un generatore di tensione variabile collegato a P8. Che cosa succede a **IN8** quando la tensione di quel pin I/O da superiore ad 1,4 V diventa inferiore? Che cosa pensate che succeda se la tensione passa da minore di 1,4V a maggiore di 1.4 V?
8. Che cosa succede al tempo di scarica di un circuito RC se il valore della resistenza diventa maggiore o minore?
9. Se volete aumentare la misura di tempo di un circuito RC in modo che sia 100 volte più lunga, quale componente? In cosa differisce questo nuovo componente dal vecchio?
10. Quali sono i nomi dei tre argomenti usati dal comando **RCTIME**? Qual è la funzione di ciascun argomento? SUGGERIMENTO: potrete trovare la risposta a questa domanda nel Manuale del BASIC Stamp.

11. Se i valori che provengono dalle misure del comando **RCTIME** non sono adeguati per il comando **PULSOUT**, potete fare due cose. La prima opzione (non consigliata) è di cambiare i componenti del circuito in maniera che vi diano i valori adeguati. Qual è l'altra opzione (raccomandata) per risolvere questo problema?
12. che cosa fa la direttiva **CON**? Spiegatelo in termini di nome e numero.

Esercizi

1. Assumiamo che in un circuito RC avete un condensatore da 0.5 µF, e volete che la misura sia 10 volte più lunga. Calcolate il valore del nuovo condensatore.
2. Data una variabile chiama **measurement**, ed un circuito RC-time collegato al pin I/O P6, scrivete un comando che misuri il tempo di scarica del circuito e lo memorizzi nella variabile.
3. Riscrivete la risposta dell'esercizio precedente in modo che nei comandi non vengano usati numeri, ma solamente alias per i valori costanti e variabili.

Progetti

1. Aggiungete un LED bicolore al circuito dell'Esercizio #4. Modificate il programma esempio in modo che il LED bicolore sia rosso quando il Servomotore ruota in senso antiorario, verde quando il Servomotore ruota in senso orario e sia spento quando il Servomotore sta fermo.
2. Aggiungete due tasti 'acceso', 'spento' al circuito dell'Esercizio #4. Modificate il programma esempio in maniera che il servomotore parta dopo la pressione ed il rilascio del tasto 'acceso' e si fermi dopo aver premuto e rilasciato il tasto 'spento'. Se premete e rilasciate il tasto 'acceso' dopo aver premuto e rilasciato il tasto 'spento', il programma deve controllare il Servomotore con il Potenziometro.
3. Costruite un circuito in cui un LED lampeggi ad una velocità proporzionale alla posizione di un Potenziometro in un circuito RC-time. Quando il Potenziometro è ruotato tutto in senso orario, il LED dovrebbe lampeggiare rapidamente, circa 10 volte al secondo. Quando il Potenziometro è ruotato tutto in senso antiorario, il LED dovrebbe lampeggiare lentamente, una volta ogni 5-10 secondi.
4. Usate **IF…THEN** per modificare il programma esempio dell'Esercizio #4 in modo che il Servomotore giri solamente con valori di **PULSOUT** tra 650 ed 850.
5. Programmate il BASIC Stamp per scalare il valore misurato dal comando **RCTIME** in modo che l'intera gamma di misure risultanti dal movimento del Potenziometro si adatti alla gamma di valori per il Servomotore del comando

PULSOUT tra 650 ed 850. SUGGERIMENTI: Per scalare la vostra misura **time** ad un valore di 200 dovrete moltiplicare usando l'operatore ***** e dividere usando l'operatore **/**. Per esempio, se viene fuori che vi serve scalare il valore di 2/5, prima moltiplicate **time** per 2, e poi dividete il risultato per 5. Dopo di che dovrete ancora aggiungere l'offset come mostrato nell'Esercizio #4.

Ulteriori Approfondimenti

In questo Capitolo sono state affrontate diverse tecniche, concetti e componenti elettronici. Alcuni dei più importanti esempi sono:

- Usare un Potenziometro come dispositivo di ingresso
- Misurare la resistenza/capacità di un dispositivo usando **RCTIME**
- Svolgere operazioni Matematiche su un valore di ingresso e trasformarlo in uscita
- Controllare un motore in base ad un valore misurato

"*Advanced Robotics: with the Toddler*", Student Workbook, Version 1.2, Parallax Inc., 2003
"*Robotics!*", Student Workbook, Version 1.5, Parallax Inc., 2000
"*SumoBot*", Student Workbook, Version 1.1, Parallax Inc., 2002

Ogni testo di robotica dello Stamps in Class usa **RCTIME** per misurare sensori resistivi per rivelare una varietà di condizioni. Ciascuna condizione porta ad operazioni matematiche ed a decisioni, il risultato finale è un movimento del robot.

"*Basic Analog and Digital*", Student Guide, Version 2.0, Parallax Inc., 2003

Basic A*nalog and Digital* usa i Potenziometri per creare una tensione variabile, il circuito viene chiamato partitore di tensione, e viene analizzato da un convertitore analogico-digitale. Un Potenziometro viene anche usato come dispositivo di ingresso per impostare la frequenza di un temporizzatore 555. Questo testo approfondisce la matematica relativa al tempo di decadimento di un circuito RC.

"*Applied Sensors*", Student Guide, Version 2.0, Parallax Inc., 2003

In questo libro, **RCTIME** viene largamente usato per raccogliere dati da una varietà di sensori.

"*Industrial Control*", Student Guide, Version 2.0, Parallax Inc., 2002

Questo libro introduce le tecniche usate estensivamente nell'industria per il controllo di macchine in base all'informazione fornita da sensori. Le tecniche ricadono nella categoria generica dei sistemi di controllo.

Capitolo #6: Visualizzatori Digitali

IL VISUALIZZATORE DIGITALE DI OGNI GIORNO

La Figura 6-1 mostra un visualizzatore sul pannello dello sportello di un forno. Quando il forno non è in funzione, esso mostra l'ora. Quando il forno è in funzione, esso mostra il temporizzatore del forno, le impostazioni di cottura, e lampeggia allo stesso ritmo di un allarme sonoro per avvisarvi della avvenuta cottura. Un microcontrollore interno allo sportello del forno rileva i tasti ed aggiorna il visualizzatore. Controlla anche i sensori interni al forno ed accende e spenge i dispositivi che regolano gli elementi riscaldanti.

Figura 6-1
Orologio Digitale e visualizzatore a 7 segmenti sullo sportello del forno

Ciascuna delle tre cifre nella Figura 6-1 si chiama display a 7 segmenti . In questo Capitolo, programmerete il BASIC Stamp per visualizzare numeri e lettere su un dispaly a 7 segmenti.

CHE COS'È UN DISPLAY A 7 SEGMENTI?

Un display a 7 segmenti è un blocchetto rettangolare composto da 7 linee di uguale lunghezza che possono essere accese selettivamente per visualizzare numeri ed alcune lettere. Un tipo molto comune è il display a 7-segmenti a LED, un blocchetto di forma rettangolare con 7 LED. La Figura 6-2 mostra il disegno pratico di un display a 7-segmenti che userete negli esercizi di questo Capitolo. C'è un LED aggiuntivo, un punto che può essere usato come punto decimale. Ciascuno dei segmenti (da A fino a G) ed il punto contengono dei LED distinti, che possono essere controllati individualmente. La maggior parte dei piedini hanno un numero a lato con una etichetta che corrisponde con un segmento LED. Il Pin 5 è etichettato DP, che significa Punto Decimale. I Pin 3 ed 8 sono etichettati "catodo comune", e saranno spiegati quando si parlerà dello schema di questo componente.

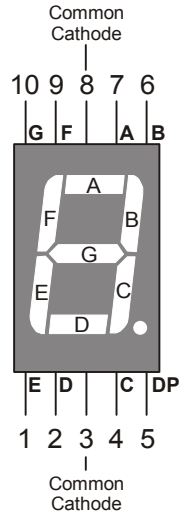



Figura 6-2
Display LED a 7-Segmenti Disegno del Componente e Piedinatura

Piedinatura: La Figura 6-2 è un esempio di piedinatura. La piedinatura contiene informazioni utili che vi aiutano a collegare un componente ad altri circuiti. La piedinatura normalmente mostra un numero per ciascun piedino, un nome per ciascun piedino ed un riferimento. Date uno sguardo alla Figura 6-2. Ciascun piedino è numerato, ed il nome di ciascun piedino indica la lettera di ciascun segmento. Il riferimento per questo componente è dato dalla sua forma. Saprete, guardandolo di fronte, che il piedino 1 è quello vicino all'angolo in basso a sinistra del display. Altri componenti hanno riferimenti meno evidenti, come la parte piana del contenitore di un LED.

La Figura 6-3 mostra lo schema elettrico dei LED all'interno di un display a 7-segmenti. Ciascun anodo dei LED è collegato ad un singolo piedino. Tutti i catodi sono internamente collegati insieme. Per il fatto che tutti i catodi sono collegati insieme, il display a 7-segmenti può essere chiamato display a "catodo comune". Collegando il piedino 3 o il piedino 8 a Vss, collegherete tutti i catodi dei LED a Vss.

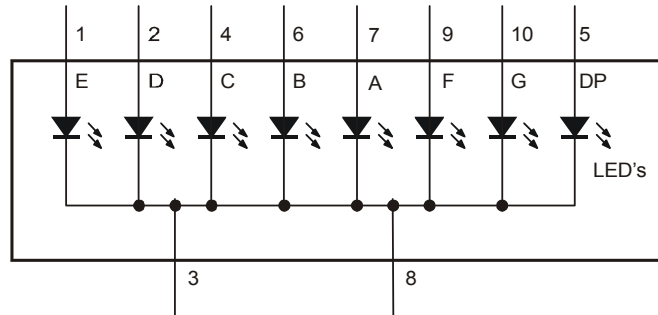



Figura 6-3
Schema Elettrico dei 7-Segmenti

6

ESERCIZIO #1: COSTRUIRE E COLLAUDARE IL DISPLAY A LED A 7-SEGMENTI

In questo Esercizio, costruirete manualmente i circuiti per collaudare ciascun segmento nel display.

Componenti per il collaudo del Display LED a 7-Segmenti

(1) Display LED a 7-segmenti
(5) Resistenze – 1 kΩ (marrone-nero-rosso)
(5) Ponticelli

Circuiti per il collaudo del Display LED a 7-Segmenti

√ Costruite i circuiti mostrati nelle Figura 6-4 e Figura 6-5 sulla vostra Board of Education o HomeWork Board dopo averle spente.
√ Riaccendete e verificate che il segmento A si illumini.

Che cosa è la x con la scritta nc sopra nello schema elettrico? La scritta nc sta per non collegato o nessun collegamento. Indica che quel particolare piedino del diplay a 7-segmenti non è collegato. Anche la x alla fine del piedino significa non collegato. Negli schemi elettrici qualche volta si usa la x e qualche volta la scritta nc.

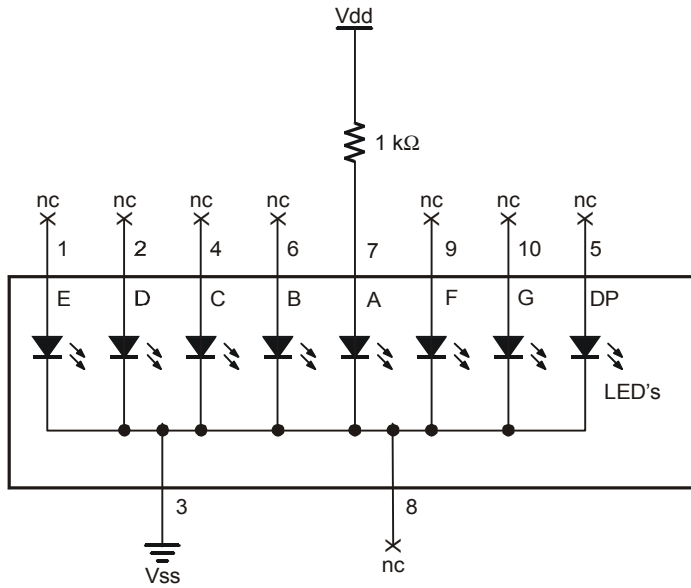



Figura 6-4
Circuito di Prova del LED del Segmento 'A' del Display.

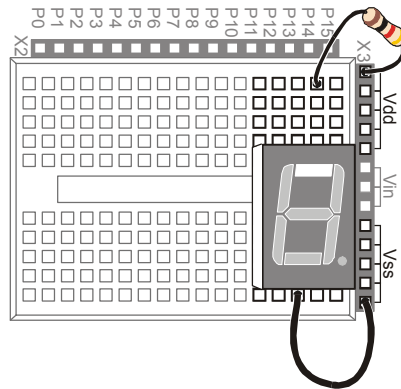



Figura 6-5
Schema pratico del Circuito di Prova per il LED del segmento 'A' del Display.

√ Spengete, e modificate il circuito collegando la resistenza al piedino del LED del segmento B come mostrato nelle Figura 6-6 e Figura 6-7.

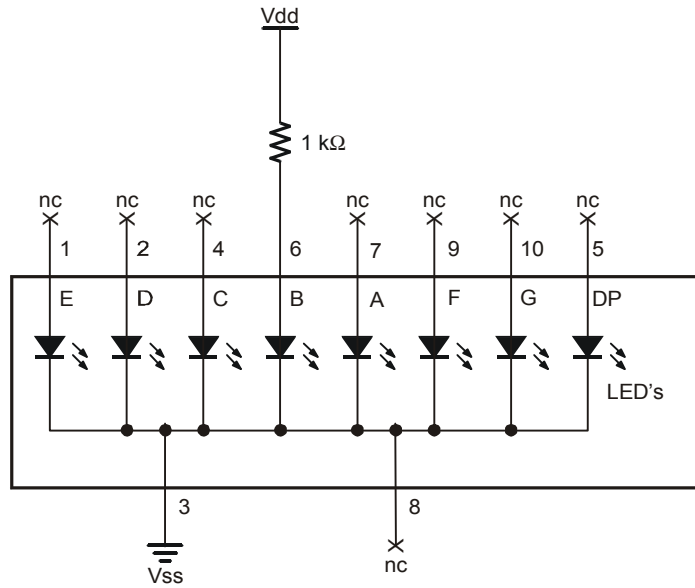



Figura 6-6
Circuito di Prova del LED del Segmento 'B' del Display.

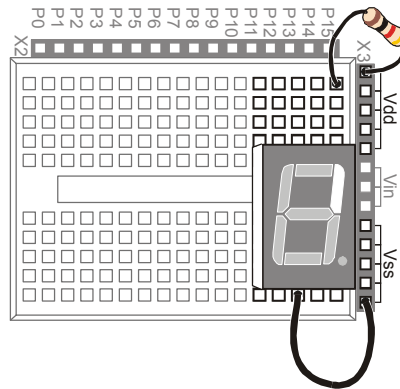



Figura 6-7
Schema pratico del Circuito di Prova per il LED del segmento 'B' del Display.

√ Riaccendete e verificate che il segmento B si illumini.

√ Usando la piedinatura dalla Figura 6-2 come guida, ripetete questi passi per i segmenti da C a G.

Il Vostro Turno – La Lettera A ed il Numero DUE

Le Figura 6-8 e Figura 6-9 mostrano la cifra '3' cablata nel display LED a 7-segmenti.

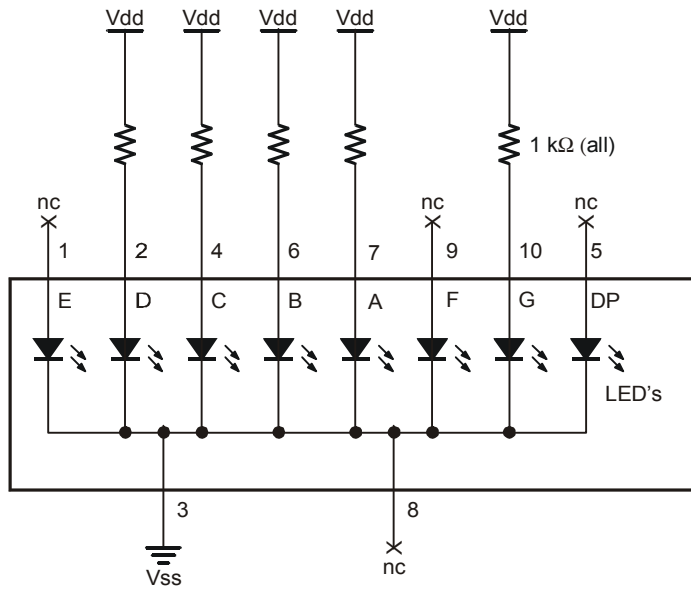



Figura 6-8
Cablaggio della cifra '3'

La cifra "3" è mostrata sul display a 7-segmenti usando il circuito di questo schema.

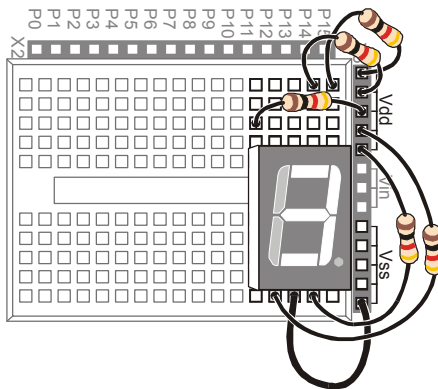



Figura 6-9
Schema di cablaggio della Figura 6-8

√ Costruite e collaudate il circuito mostrato nelle Figura 6-8 e Figura 6-9, e verificate che venga visualizzato il numero TRE.

√ Disegnate lo schema del numero DUE.

√ Costruite e collaudate il circuito per assicurarvi che funzioni. Fate ricerca guasti se necessario.

√ Ripetere per la lettera 'A'.

ESERCIZIO #2: CONTROLLARE IL DISPLAY LED A 7-SEGMENTI

In questo Esercizio, collegherete il display LED a 7-segmenti al BASIC Stamp, e quindi lancerete un programma semplice per collaudare ed assicurarvi che ciascun LED sia connesso correttamente.

Componenti del Display LED a 7-Segmenti

(1) Display LED a 7-segmenti
(8) Resistenze – 1 kΩ (marrone-nero-rosso)
(5) Ponticelli

Collegare il Display LED a 7-Segmenti al BASIC Stamp

La Figura 6-10 mostra lo schema elettrico e la Figura 6-11 mostra il cablaggio per questo esempio di display LED a 7-segmenti controllato dal BASIC Stamp.

√ Costruite il circuito mostrato nelle Figura 6-10 e Figura 6-11.

Schema elettrico e piedinatura: se state provando a costruire il circuito dello schema di Figura 6-10 senza guardare la Figura 6-11, consultate almeno la piedinatura del display LED a 7-segmenti (Figura 6-2, pagina 170).

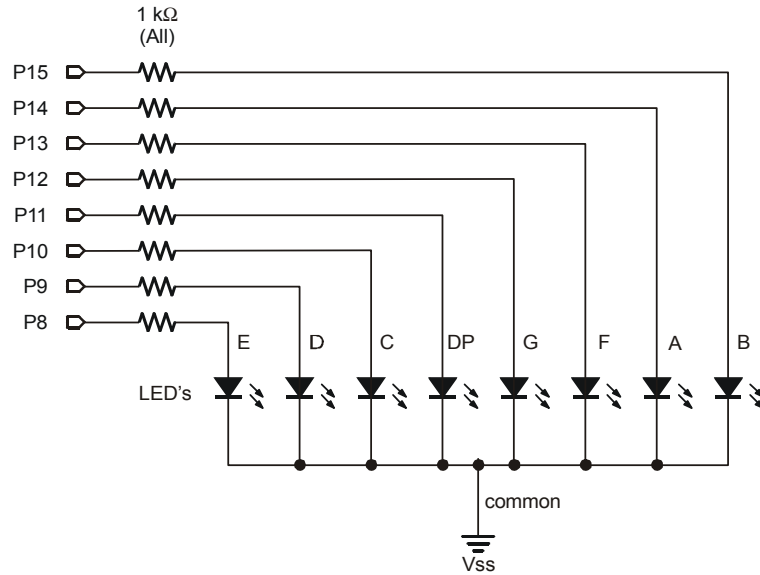



Figura 6-10
Schema elettrico del Display LED a 7-Segmenti collegato al BASIC Stamp

Fate attenzione alle Resistenze collegate a P13 e P14. Guardate attentamente alle Resistenze collegate a P13 e P14 in Figura 6-11. C'è uno spazio tra queste due Resistenze. Lo spazio c'è perché il pin 8 del display LED a 7-segmenti è lasciato scollegato. Una Resistenza collega il pin I/O P13 al pin 9 del display LED a 7-segmenti. Un'altra Resistenza collega P14 al pin 7del display LED a 7-segmenti.

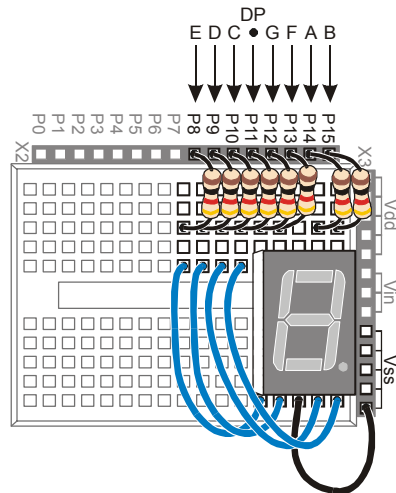



Figura 6-11
Schema di cablaggio della Figura 6-10

Usate le lettere dei segmenti su questo schema come riferimento.

Dispositivi Paralleli: Il display LED a 7-segmenti viene chiamato dispositivo parallelo perché dovete usare più di una linea I/O alla volta per inviargli dati (segnali alti e bassi) che gli dicano cosa fare. Nel caso di questo display LED a 7-segmenti, ci vogliono pin 8 I/O per far funzionare il dispositivo.

Bus Parallelo : I fili che trasmettono i segnali **ON/OFF** dal BASIC Stamp al display LED a 7-segmenti vengono chiamati bus parallelo. Notate che questi fili nella Figura 6-10 sono disegnati come linee parallele. Il termine parallelo diventa sensato data la geometria dello schema elettrico.

Programmazione della prova del display LED a 7-segmenti.

I comandi **HIGH** e **LOW** accettano una variabile come argomento pin. Per provare ciascun segmento, uno per volta, semplicemente mettete i comandi **HIGH** e **LOW** in un ciclo **FOR…NEXT**, e usate l'indice per impostare il pin I/O, alto quindi di nuovo basso.

√ Digitate e lanciate SegmentTestWithHighLow.bs2.
√ Verificate che ogni segmento nel display LED a 7-segmenti lampeggi brevemente, accendendosi e spengendosi.
√ Fare una lista di quale segmento viene acceso da quale pin I/O.

Programma di Esempio: SegmentTestWithHighLow.bs2

```
' Che cosa è un Microcontrollore- SegmentTestWithHighLow.bs2
' In un display LED a 7-segmenti prova individualmente ciascun segmento.
```

```
'{$STAMP BS2}
'{$PBASIC 2.5}

pinCounter     VAR     Nib

DEBUG "I/O Pin", CR,
      "-------", CR

FOR pinCounter = 8 TO 15

  DEBUG DEC2 pinCounter, CR
  HIGH pinCounter
  PAUSE 1000
  LOW pinCounter

NEXT
```

Il Vostro Turno – uno schema diverso

La rimozione del comando **`LOW pinCounter`** avrà un effetto interessante:

- √ Apostrofate il comando **`LOW pinCounter`** aggiungendo un apostrofo a sinistra.
- √ Lanciate il programma modificato ed osservate l'effetto.

ESERCIZIO #3: VISUALIZZARE CIFRE

Se includete il punto decimale ci sono otto diversi pin I/O del BASIC Stamp che inviano segnali ON/OFF al display LED a 7-segmenti. Il che significa otto diversi comandi **`HIGH`** o **`LOW`** solo per visualizzare un numero. Se volete contare da zero a nove, ne risulterebbe un programma molto grande. Fortunatamente, ci sono variabili speciali che potete usare per impostare i valori alti e bassi per gruppi di pi I/O.

In questo Esercizio, per controllare i segnali ON/OFF inviati dai pin del BASIC Stamp userete numeri binari ad 8 cifre invece di comandi **`HIGH`** e **`LOW`**. Impostando le variabili speciali chiamate **`DIRH`** e **`OUTH`** uguali ai numeri binari, con un singolo comando potrete controllare i segnali ON/OFF inviati da tutti i pi I/O collegati al circuito del display LED a 7-segmenti.

8 bit: Un numero binario che abbia 8 cifre è detto avere otto bit. Ciascun bit è una casella dove potete memorizzare un 1 o uno 0.

Un byte è una variabile che contiene 8 bit. Ci sono 256 diverse combinazioni di zero e uno che usando 8 bit potete usare per contare da 0 a 255. Questa è la ragione per cui con un byte potete memorizzare un numero da 0 a 255.

Componenti e Circuiti per Visualizzare Cifre

Gli Stessi dell'Esercizio precedente

Programmazione dei Gruppi ON/OFF per l'uso dei Numeri Binari

6

In questo Esercizio, farete esperimenti con le variabili **DIRH** e **OUTH**. **DIRH** è una variabile che controlla la direzione (ingresso o uscita) dei pin I/O da P8 fino a P15. **OUTH** controlla i segnali alti e bassi che ciascun pin I/O invia. Come potrete presto vedere, **OUTH** è particolarmente utile perché con un solo comando potete impostare i segnali ON/OFF per otto diversi pin alla volta. Di seguito c'è un programma esempio che mostra come possono essere usate queste due variabili per visualizzare il conteggio da 0 a 9 sul display LED a 7-segmenti senza usare i comandi **HIGH** e **LOW**:

Programma di Esempio: DisplayDigits.bs2

Questo programma esempio alternerà sul display LED a 7-segmenti le cifre da 0 a 9.

√ Digitate e lanciateDisplayDigits.bs2.
√ Verificate che vengano visualizzate le cifre da 0 a 9.

```
' Che cosa è un Microcontrollore- DisplayDigits.bs2
' Visualizza le cifre da 0 a 9 sul display LED a 7-segmenti.

'{$STAMP BS2}
'{$PBASIC 2.5}

OUTH = %00000000                     ' OUTH initializzato basso.
DIRH = %11111111                     ' Imposta P8-P15 come tutte uscite basse.
                                     ' Cifre:
'        BAFG.CDE
OUTH = %11100111                     ' 0
PAUSE 1000
OUTH = %10000100                     ' 1
PAUSE 1000
OUTH = %11010011                     ' 2
PAUSE 1000
OUTH = %11010110                     ' 3
```

```
PAUSE 1000
OUTH = %10110100                         ' 4
PAUSE 1000
OUTH = %01110110                         ' 5
PAUSE 1000
OUTH = %01110111                         ' 6
PAUSE 1000
OUTH = %11000100                         ' 7
PAUSE 1000
OUTH = %11110111                         ' 8
PAUSE 1000
OUTH = %11110110                         ' 9
PAUSE 1000

DIRH = %00000000                         ' Impostazione pin I/O come ingressi,
                                         ' segmenti spenti.
END
```

Come funziona DisplayDigits.bs2

La Figura 6-12 mostra come potete usare le variabili **DIRH** e **OUTH** per controllare la direzione e lo stato (ON/OFF) dei pin I/O da P8 fino a P15.

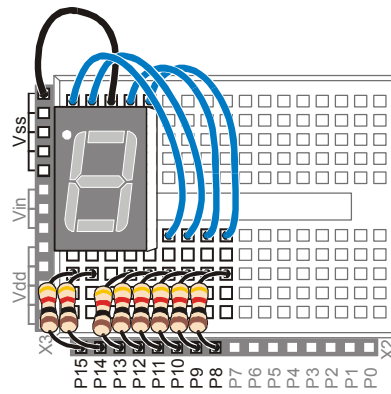



Figura 6-12
Uso di DIRH e OUTH per impostare tutti i pin I/O come uscite basse

Il primo comando:

```
OUTH = %00000000
```

Rende pronti all'invio dei segnali bassi tutti i pin I/O (da P8 fino a P15). Se vengono inviati tutti segnali bassi, tutti i segmenti del display LED a 7-segmenti si spengeranno. Se volete che tutti i pin di I/O inviino i segnali alti, potreste invece usare **`OUTH = %11111111`**.

Che cosa fa il simbolo %? Il simbolo % viene usato per dire all'Editor del BASIC Stamp che quel numero è un numero binario. Per esempio, il numero binario %00001100 è lo stesso del numero decimale 12. Come vedrete in questo Esercizio, i numeri binari possono facilitare molti compiti di programmazione.

I segnali bassi non verranno inviati ai pin I/O fino a che non userete la variabile **`DIRH`** per cambiare tutti i pin I/O da ingressi ad uscite. Il comando:

```
DIRH = %11111111
```

Imposta come uscite tutti i pin I/O da P8 fino a P15. Nel momento in cui questo comando viene eseguito, tutti i pin da P8 fino a P15 cominciano ad inviare i segnalo basso. Questo perché subito prima del comando **`DIRH`** è stato eseguito il comando **`OUTH = %00000000`**. Dal momento che il comando **`DIRH`** imposta tutti i pin I/O come uscite, iniziano ad inviare i loro segnali bassi. Potreste anche usare il comando **`DIRH = %00000000`** per ricambiare tutti i pin I/O come ingressi.

Prima che i pin I/O diventino uscite: fino al momento in cui i pin I/O vengono cambiati da ingresso ad uscite, essi rimangono in attesa di un segnale per aggiornare la variabile **`INH`**. Questa è la variabile che contiene **`IN8`**, **`IN9`**, su fino ad **`IN15`**. Queste variabili possono essere usate nello stesso modo in cui **`IN3`** ed **`IN4`** sono state usate per rilevare lo stato dei tasti nel Capitolo #3.

Tutti i pin I/O del BASIC Stamp partono come ingressi. Questo viene chiamato "default". Prima che un pin I/O del BASIC Stamp inizi ad inviare un segnale alto o basso, dovete dirgli di diventare un'uscita. Ambedue i comandi **`HIGH`** e **`LOW`** cambiano automaticamente in uscita la direzione di un pin I/O del BASIC Stamp. Anche mettere un 1 nella variabile **`DIRH`** fa diventare un pin I/O un pin di uscita.

La Figura 6-13 mostra come usare la variabile **`OUTH`** per mandare specifici segnali alti o bassi ai pin P8 fino a P15. Un 1 binario viene usato per inviare un segnale alto, ed uno 0 binario per inviare un segnale basso. Questo esempio visualizza il numero tre sul display LED a 7-segmenti:

```
'        BAFG.CDE
OUTH = %11010110
```

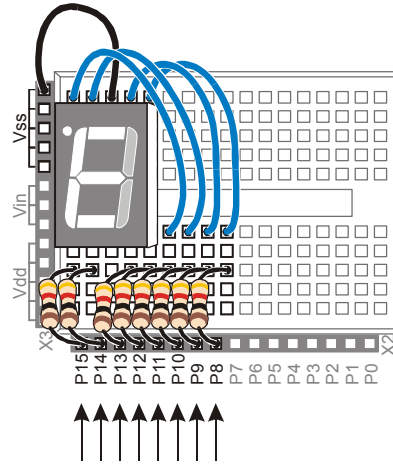



Figura 6-13
Usare OUTH per controllare i segnali ON/OFF di P8 – P15.

Il display è disposto in modo che il numero tre sia visualizzato capovolto perché diviene più chiaro come il valore nella riga **OUTH** sia corrispondente ai pin I/O. Il comando **OUTH = %11010110** usa zero binari per impostare bassi i pin I/O P8, P11, e P13, ed usa uno binari per impostare alti i pin P9, P10, P12, P14 e P15. La riga subito prima del comando è un commento che indica le etichette dei segmenti allineate con i valori binari che accendono o spengono quel segmento. Il prossimo programma esempio mostra come impostare i numeri binari del comando **OUTH** per far visualizzare al display LED a 7-segmenti le cifre da zero a nove.

Dentro i comandi HIGH e LOW: Il comando **HIGH 15** in realtà ha lo stesso significato di **OUT15 = 1** seguito da **DIR15 = 1**. Similmente, il comando **LOW 15** è lo stesso di **OUT15 = 0** seguito da **DIR15 = 1**. Se volete reimpostare P15 come ingresso, usate **DIR15 = 0**. potrete quindi usare **IN15** per rilevare (invece di inviare) segnali ON/OFF.

Il Vostro Turno – Visualizzare da A fino ad F

√ Immaginate quale configurazione di bit (combinazione di zero ed uno) vi servirà per visualizzare le lettere A, b, C, d, E, ed F.

√ Modificate SevenSegment0to9 in modo che visualizzi A, b, C, d, E, F.

Confronto fra Decimale ed Esadecimale Le cifre base del sistema numerico decimale (base-10) sono: 0, 1, 2, 3, 4, 5, 6, 7, 8, e 9. Nel sistema numerico esadecimale (base-16) le cifre base sono: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, F. La Base-16 viene ampiamente usata nella programmazione sia dei computer che dei microcontrollori. Dal momento che avete già indovinato come visualizzare i caratteri da A fino ad F, potete modificare ulteriormente il vostro programma per contare in esadecimale da 0 ad F.

Mantenere Listati dei Gruppi ON/OFF

Il comando **LOOKUP** facilita la scrittura del codice dei gruppi per la visualizzazione sul display LED a 7-segmenti. Il comando **LOOKUP** vi consente di trovare un elemento di una lista. Di seguito viene mostrato un esempio di codice che usa il comando **LOOKUP**:

```
LOOKUP index, [7, 85, 19, 167, 28], value
```

In questo comando ci sono due variabili, **index** e **value**. Se **index** è 0, **value** memorizza il **7**. Se **index** è **1**, **value** memorizza **85**. dal momento che **index** è 2, in questo esempio, il comando **LOOKUP** mette **19** in **value**, e questo è ciò che visualizza il Terminale di Debug.

Programma di Esempio: SimpleLookup.bs2

√ Caricate e Lanciate SimpleLookup.bs2.
√ Lanciate il programma così com'è, con la variabile **index** impostata uguale a 2.
√ Provate ad impostare la variabile **index** con i numeri da 0 a 4.
√ Rilanciate il programma dopo ciascun cambio della variabile **index** ed annotate quale valore della lista viene messo nella variabile **value**.
√ Opzione: Modificate il programma inserendo il comando **LOOKUP** in un ciclo **FOR…NEXT** che conti da 0 a 4.

```
' Che cosa è un Microcontrollore- SimpleLookup.bs2
' Visualizza un valore usando un index ed una tabella di lookup.

' {$STAMP BS2}
' {$PBASIC 2.5}

value          VAR     Byte
index          VAR     Nib

index = 2
```

```
DEBUG ? index

LOOKUP index, [7, 85, 19, 167, 28], value

DEBUG ? value, CR

DEBUG "Cambiate la variabile index con un ", CR,
      "numero diverso (tra 0 e 4).", CR, CR,

      "Lanciate il programma modificato", CR,
      "Controllate per vedere quale numero", CR,
      "il comando LOOKUP mette nella", CR,
      "variabile value."

END
```

Programma di Esempio: DisplayDigitsWithLookup.bs2

Questo programma esempio mostra come può essere realmente utile il comando **LOOKUP** per memorizzare il gruppo di bit usati nella variabile **OUTH**. Ancora, l'**index** viene usato per scegliere quale valore binario mettere nella variabile **OUTH**. Questo programma esempio conta da 0 a 9. La differenza tra questo programma e DisplayDigits.bs2 è che questo programma è molto più versatile. È molto più veloce e facile regolare perché differenti sequenze di numeri usino le tabelle di lookup.

- √ Caricate e Lanciate DisplayDigitsWithLookup.bs2.
- √ Verificate che faccia le stesse cose del programma precedente(con molto meno lavoro).
- √ Mentre il programma gira, osservate il Terminale di Debug. Esso mostra come viene usato il valore di **index** dal comando **LOOKUP** per caricare dalla lista il valore binario giusto in **OUTH**.

```
' Che cosa è un Microcontrollore- DisplayDigitsWithLookup.bs2
' Usa una tabella di lookup per memorizzare e visualizzare cifre sul display
' LED a 7 segmenti.

'{$STAMP BS2}
'{$PBASIC 2.5}

index          VAR     Nib

OUTH = %00000000
DIRH = %11111111

DEBUG "index  OUTH   ", CR,
      "-----  --------", CR
```

```
FOR index = 0 TO 9

  LOOKUP index, [ %11100111, %10000100, %11010011,
                  %11010110, %10110100, %01110110,
                  %01110111, %11000100, %11110111, %11110110 ], OUTH

  DEBUG "   ", DEC2 index, "  ", BIN8 OUTH, CR

  PAUSE 1000

NEXT

DIRH = %00000000

END
```


Il Vostro Turno – un nuovo modo di visualizzare da 0 ad F

√ Modificate DisplayDigitsWithLookup.bs2 in modo che conti da 0 ad F in esadecimale. Non dimenticate di aggiornare i valori di **`start`** e di **`end`** del ciclo **`FOR…NEXT`**.

ESERCIZIO #4: VISUALIZZARE LA POSIZIONE DI UNA MANOPOLA

Nell'Esercizio #4 del Chapter #5, avete usato il Potenziometro per controllare la posizione di un Servomotore. In questo Esercizio, visualizzerete la posizione della manopola del Potenziometro usando il display LED a 7 segmenti.

Componenti dell'Indice e del display

(1) 7-segment LED display
(8) Resistenze – 1 kΩ (marrone-nero-rosso)
(1) Potenziometro – 10 kΩ
(1) Resistenza – 220 Ω (rosso-rosso-marrone)
(1) Condensatore – 0.1 µF
(7) Ponticelli

Costruzione dell'Indice e del display

La Figura 6-14 mostra lo schema elettrico del circuito del Potenziometro che deve essere aggiunto al progetto. La Figura 6-15 mostra lo schema di cablaggio del circuito della Figura 6-14 combinato con il circuito della Figura 6-10 a Pagina 1766.

√ Aggiungete il circuito del Potenziometro al circuito del display LED a 7-segmenti come mostrato nella Figura 6-15.

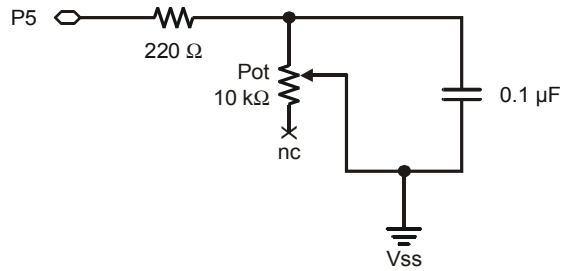



Figura 6-14
Schema elettrico del circuito del Potenziometro aggiunto al Progetto

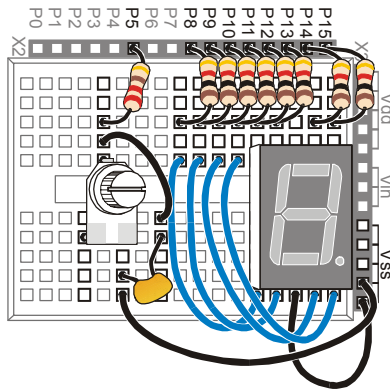



Figura 6-15
Schema di cablaggio del Progetto

Programmazione dell'Indice e del display

C'è un comando utile chiamato **LOOKDOWN**, ebbene sì, è l'opposto del comando **LOOKUP**. Mentre il comando **LOOKUP** vi restituisce un numero in base all'indice, il comando **LOOKDOWN** vi restituisce un indice in base ad un numero.

Programma di Esempio: SimpleLookdown.bs2

Questo programma esempio mostra come funziona il comando **LOOKDOWN**.

√ Caricate e Lanciate SimpleLookdown.bs2.
√ Lanciate il programma così com'è, con la variabile **value** impostata uguale a 167, ed usate il Terminale di Debug per osservare il valore di **index**.

√ Provate ad impostare la variabile **value** uguale a ciascuno dei numeri elencati dal comando **LOOKDOWN**: 7, 85, 19, 28.

√ Rilanciate il programma dopo ciascuna variazione della variabile **value** ed annotate quale valore dell'elenco viene messo nella variabile **index**.

A meno che non dichiarate un modo diverso di confronto, il comando **LOOKDOWN** controlla per vedere se un valore è uguale ad uno presente nell'elenco. Potrete anche controllare se un valore è maggiore di, minore o uguale, etc. Per esempio, per vedere se un valore è minore od uguale al valore della variabile, usate l'operatore **<=** subito prima della prima parentesi che inizia l'elenco.

√ Modificate SimpleLookdown.bs2 sostituendo **value** e la dichiarazione **LOOKDOWN**:

```
value = 35
LOOKDOWN value, <= [ 7, 19, 28, 85, 167 ], index
```

√ Sperimentate con valori diversi e vedete se la variabile indice visualizza quello che vi sareste aspettato.

Domanda Subdola: Che cosa succede se il vostro valore è maggiore di 167? Questa piccola imprecisione nel comando **LOOKDOWN** può causare problemi perché il comando **LOOKDOWN** non effettua alcuna variazione all'indice.

```
' Che cosa è un Microcontrollore- SimpleLookdown.bs2
' Scegliere un indice usando un valore ed una tabella di lookup.

' {$STAMP BS2}
' {$PBASIC 2.5}

value          VAR     Byte
index          VAR     Nib

value = 167

DEBUG ? value

LOOKDOWN value, [7, 85, 19, 167, 28], index

DEBUG ? index, CR

DEBUG "Cambiate la variabile index con un ", CR,
      "numero diverso del seguente elenco.", CR, CR,
```

```
      "7, 85, 19, 167, or 28.", CR, CR,

END
```

Programma di Esempio: DialDisplay.bs2

Questo esempio rispecchia la posizione della manopola del Potenziometro accendendo i segmenti esterni del display LED a 7-segmenti come mostrato in Figura 6-16.

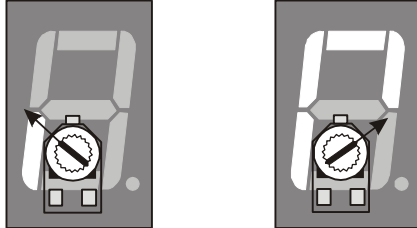



Figura 6-16
Visualizzazione della posizione del Potenziometro con il Display LED a 7-Segmenti

- √ Caricate e Lanciate DialDisplay.bs2.
- √ Ruotate il potenziometro ed assicuratevi che funzioni.
- √ Quando lanciate il programma esempio, potrebbe non essere preciso come mostrato in Figura 6-16. Regolate i valori della tabella lookdown in maniera che il display digitale rappresenti nel modo più accurato possibile la posizione del Potenziometro.

```
' Che cosa è un Microcontrollore- DialDisplay.bs2
' Visualizza la posizione del Potenziometro usando il display LED 7-segmenti.

'{$STAMP BS2}
'{$PBASIC 2.5}

index          VAR     Nib
time           VAR     Word

OUTH = %00000000
DIRH = %11111111

DO

  HIGH 5
  PAUSE 100
  RCTIME 5, 1, time

  LOOKDOWN time, <= [40, 150, 275, 400, 550, 800], index
```

```
  LOOKUP index, [ %11100101, %11100001, %01100001,
                  %00100001, %00000001, %00000000 ], OUTH

LOOP
```

Come Funziona DialDisplay.bs2

Questo programma esempio esegue una misura **`RCTIME`** del Potenziometro e la memorizza nella variabile denominata **`time`**.

```
HIGH 5
PAUSE 100
RCTIME 5, 1, time
```

La variabile **`time`** viene quindi usata nella tabella di **`LOOKDOWN`**. La tabella di **`LOOKDOWN`** decide quale numero dell'elenco è 'minore di', quindi carica nella variabile **`index`** il numero scelto (in questo caso da 0 a 5).

```
LOOKDOWN time, <= [40, 150, 275, 400, 550, 800], index
```

Quindi, la variabile **`index`** viene usata da una tabella di **`LOOKUP`** per scegliere il numero binario da caricare nella variabile **`OUTH`**.

```
LOOKUP index, [ %11100101, %11100001, %01100001,
                %00100001, %00000001, %00000000 ], OUTH
```

Il Vostro Turno – Aggiungere un Segmento

DialDisplay.bs2 accende solamente cinque dei sei segmenti esterni quando girate la manopola. Su DialDisplay.bs2 la sequenza di accensione dei LED è E, F, A, B, C, ma non D.

√ Salvate DialDisplay.bs2 con il nuovo nome DialDisplayYourTurn.bs2.
√ Modificate DialDisplayYourTurn.bs2 in modo che faccia accendere in sequenza tutti e sei i segmenti esterni quando il potenziometro viene ruotato. La sequenza dovrebbe essere: E, F, A, B, C, e D.

SOMMARIO

Questo Capitolo ha spiegato il display LED a 7-segmenti, e come leggere la piedinatura di un componente complesso. Questo Capitolo ha inoltre spiegato alcune tecniche per dispositivi e circuiti che hanno ingressi paralleli. Sono state spiegate le variabili **DIRH** ed **OUTH** come metodo per controllare il valore dei pin I/O da P8 a P15 del BASIC Stamp. Sono stati spiegati i comandi **LOOKUP** e **LOOKDOWN** come modo di riferimento ad elenchi di valori usati per visualizzare lettere e numeri.

Domande

1. In un display LED a 7-segmenti, qual'è il componente attivo che rende il display leggibile quando un microcontrollore invia segnali alti o bassi?
2. Qual è la differenza tra simbolo schematico e piedinatura? Quale relazione c'è fra le due cose?
3. Che cosa significa catodo comune? Che cosa pensate voi che significhi catodo comune?
4. Qual è il gruppo di fili che conducono segnali da e verso un dispositivo parallelo e come si chiama?
5. Come funzionano le variabili **DIRH** ed **OUTH**?
6. Come si chiamano i comandi usati in questo Capitolo per gestire elenchi di valori?
7. Quale carattere dovete digitare per inserire un numero binario nell'Editor del BASIC Stamp?
8. Come effettuate una comparazione "minore di od uguale a"?

Esercizi

1. Scrivete un comando **OUTH** per impostare P8, P10, P12 alti e P9, P11, P13 bassi. Assumendo che tutti i vostri pin di I/O iniziano come ingressi, scrivete il comando **DIRH** in modo che faccia inviare segnali ON/OFF ai pin I/O lasciando P14, P15 configurati come ingressi.
2. Modificate una linea di SegmentTestWithHighLow.bs2 dall'Esercizio #2 in maniera che provi un display LED a 7-segmenti collegato a P0 - P7. Disegnate lo schema elettrico del vostro progetto modificato.
3. Assumete di spostare tutto il circuito del display LED a 7-segmenti in modo che invece di essere collegato a P8 - P15, sia ora collegato a P0 - P7. La variabile che indirizza questi pin I/O è **OUTL**. La L sta per byte basso mentre la H di **OUTH** sta per byte alto. Riscrivete il comando **LOOKUP** in DisplayDigitsWithLookup.bs2 in

modo che indirizzi la variabile **OUTL**. Riscrivete gli altri cinque comandi nel programma che devono essere cambiati per assecondare lo spostamento del circuito del display LED a 7-segmenti dal byte superiore dei pin I/O del BASIC Stamp al byte inferiore.

4. Modificate il comando **LOOKDOWN** in SimpleLookdown.bs2 così che funzioni con l'operatore **>=**. Questo operatore significa "maggiore di od uguale a". Suggerimento: perché questa modifica funzioni correttamente, dovrete modificare l'ordine dei valori nell'elenco.
5. Scrivete i valori di **OUTH** necessari per far accendere le lettere: a, C, d, F, H, I, n, P, S.

Progetti

1. Compitate "FISH CHIPS e dIP" più volte con il display LED a 7-segmenti. Fate accendere ciascuna lettera per 400 ms.
2. Modificate DialDisplay.bs2 dell'Esercizio #4 così che faccia comportare il display LED a 7-segmenti come mostrato in Figura 6-17. L'immagine sulla sinistra mostra come dovrebbe accendersi il display LED a 7-segmenti quando il Potenziometro è al centro, con acceso solamente il segmento G. L'immagine di centro della Figura 6-17 mostra accesi i segmenti E e D. Come il Potenziometro viene ruotato in senso antiorario, i segmenti E, D, e C si accendono in sequenza. L'immagine di destra della Figura 6-17 mostra la manopola del Potenziometro ruotata in senso orario dalla posizione centrale con i segmenti F ed A accesi. Quando la manopola del Potenziometro viene ruotata in senso orario (dalla posizione centrale), si dovranno accendere in sequenza i segmenti F, A, e B.

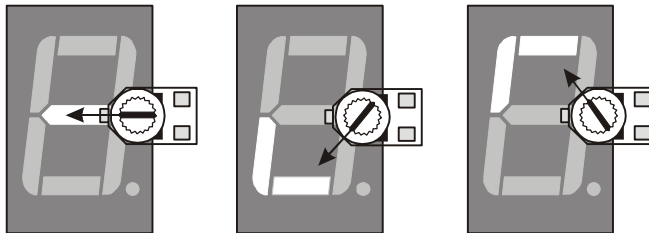



Figura 6-17
Visualizzazione dello spostamento della posizione del Potenziometro dal Centro

La manopola del Potenziometro nella posizione centrale (sinistra), ruotato in senso antiorario (centro), e ruotato in senso orario (destra).

3. Progettate un temporizzatore a scalare da 59-secondi che possa essere inizializzato da tasti. Questo temporizzatore può essere caricato con un valore in secondi (tra 1 e 59). Usate due tasti per caricare il numero dei secondi. Un tasto con la sua pressione e rilascio deve scorrere tra i valori da 0 a 9. L'altro tasto deve essere usato per selezionare il valore corrente visualizzato sul display LED a 7-segmenti. Dopo aver inserito i due valori, per iniziare il conteggio premere e tenere premuti brevemente ambedue i tasti. Il display deve mostrare ciascuna cifra del tempo rimanente per ½ secondo ogni due secondi. Il punto sul display deve lampeggiare ogni secondo per avvertire che il temporizzatore sta effettuando il conteggio. Quando il conteggio è finito, tutti i segmenti LED devono lampeggiare rapidamente per 5 secondi.

Ulteriori Approfondimenti

Così come per tutti i testi citati in questa sezione, anche questo è disponibile per essere scaricato gratuitamente dal sito www.parallax.com.

"StampWorks", Workbook, Version 1.2, Parallax Inc., 2001

StampWorks è scritto dall'autore della rivista Nuts & Volts John Williams, ed espone una gran varietà di esperimenti. Questo testo è una collezione di 32 esperimenti con inclusi svariati usi del display LED a 7-segmenti ed altri tipi di display. Questo testo fa uso della scheda INEX 1000, che ha inseriti molti dei componenti che avete usato in questo testo.

Capitolo #7: Misurazione della Luce

DISPOSITIVI CHE CONTENGONO SENSORI DELLA LUCE

Avete già lavorato con due diversi tipi di sensori. Il tasto può essere pensato come un semplice sensore di pressione, ed il Potenziometro che è un sensore di posizione e/o rotazione. Ci sono molti generi di sensori inseriti nelle applicazioni e nelle macchine che non sono cosi ovvi come il tasto o la manopola. Altri sensori comuni misurano cose tipo temperatura, fumo, vibrazioni, inclinazione e luce. Sebbene ciascuno di questi diversi tipi di sensori possono essere trovati in molti dei dispositivi che la maggior parte delle persone, usano giornalmente, i sensori di luce sono probabilmente i più comuni.

Un esempio di dispositivo di uso giornaliero che contiene un sensore di luce è probabilmente il vostro televisore. Se può essere controllato da un telecomando, possiede al suo interno un rivelatore di un tipo di luce chiamata infrarosso che non può essere vista dall'occhio umano. Il telecomando usa la luce infrarossa per trasmettere informazioni circa il canale, volume, ed altri tasti che potete premere per controllare la TV. Un altro esempio comune è la telecamera digitale. Il sensore di luce della telecamera aiuta a regolare l'esposizione per varie condizioni di illuminazione così che l'immagine sia chiara senza essere influenzata dal fatto che sia una giornata soleggiata o nuvolosa.

CONOSCIAMO LA FOTORESISTENZA

I sensori di luce hanno molti impieghi diversi, ed esistono con diverse forme, dimensioni, e con prezzi differenti. Alcuni sensori sono progettati per rilevare un particolare colore di luce, come il blu, verde, rosso, o infrarosso. Alcuni sensori non sono sensibili al colore della luce perché reagiscono all'intensità della luce. Altri sensori reagiscono solamente a tipi speciali di luce emessa da reazioni chimiche. I sensori di luce hanno inoltre una quantità di modi per indicare ad un microcontrollore che cosa vedono. Alcuni sensori inviano una tensione, alcuni inviano una sequenza di valori binari, ed altri reagiscono ai diversi tipi di luce o livelli di luce cambiando la resistenza.

Dei sensori di luce che reagiscono alla luce cambiando la loro resistenza, la fotoresistenza mostrata in Figura 7-1 è probabilmente la più comune, la meno costosa e facile da usare. Il suo ingrediente attivo è un composto chimico chiamato solfuro di cadmio (CdS). Questo composto cambia resistenza in rapporto alla quantità di luce che arriva sulla superficie del dispositivo. Una luce brillante causa una bassa resistenza ai capi mentre una bassa luminosità causa valori di resistenza più alti.

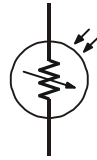



Figura 7-1
Schema elettrico della Fotoresistenza e disegno del componente.

Il rivestimento superficiale di solfuro di cadmio della fotoresistenza è sulla parte superiore del disegno del componente.

Come con il Potenziometro, potete usare una fotoresistenza in un circuito RC-time. Il programma per leggere il valore della fotoresistenza è circa uguale a quello per leggere il valore del Potenziometro. Sebbene la programmazione sia la stessa, la luce è molto diversa dalla rotazione o dalla posizione. Gli esercizi in questo Capitolo si focalizzano sulle applicazioni che usano la luce (invece della posizione) per fornire informazioni al microcontrollore. In seguito saranno introdotte alcune tecniche di programmazione PBASIC, per aiutarvi per la memorizzazione di dati a lungo termine, e per rendere più maneggiabili e facilmente leggibili i vostri programmi.

ESERCIZIO #1: COSTRUIRE E COLLAUDARE IL MISURATORE DI LUCE

In questo Esercizio, costruirete e collauderete un circuito RC-time che legge il valore di una fotoresistenza. La misura RC-time vi darà un'idea del livello di luce rilevato dalla superficie sensibile della fotoresistenza. Come per il collaudo del potenziometro, il tempo misurato dal comando **RCTIME** sarà visualizzato nel Terminale di Debug.

Componenti per il Collaudo del Misuratore di Luce

(1) Fotoresistenza
(1) Resistenza – 220 Ω (rosso-rosso-marrone)
(1) Condensatore – 0.01 μF
(1) Condensatore – 0.1 μF
(1) Ponticello

Sebbene nell'elenco dei componenti ci siano due condensatori, in ogni momento ne userete nel circuito solamente uno.

Costruzione del Circuito RC Time con una Fotoresistenza

La Figura 7-2 mostra lo schema elettrico del circuito RC-time che userete in questo Capitolo, e la Figura 7-3 mostra lo schema di cablaggio. Questo circuito è diverso dal circuito del Potenziometro del Chapter #5, Esercizio #3 in due aspetti. Primo, il pin I/O usato per misurare il tempo di decadimento è diverso (P2). Secondo, la variabile Resistenza è ora una fotoresistenza invece di un Potenziometro.

√ Costruite il circuito mostrato nella Figura 7-2 e nella Figura 7-3.

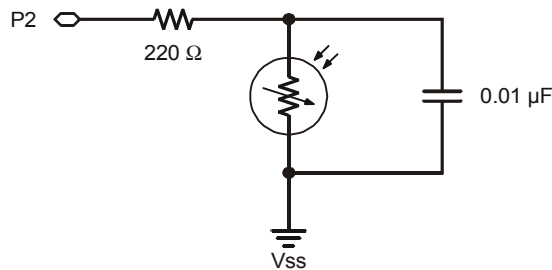



Figura 7-2
Schema elettrico del circuito RC-time della Fotoresistenza

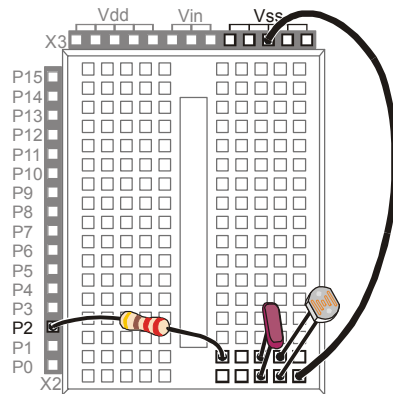



Figura 7-3
Schema di cablaggio per la Figura 7-2

Programmazione per il collaudo della Fotoresistenza

Il primo programma esempio (TestFotoresistenza.bs2) è solamente una versione modificata di ReadPotWithRcTime.bs2 del Chapter #5, Esercizio #3. Il circuito del Potenziometro del Chapter #5 era collegato al pin I/O P7. Il circuito in questo Esercizio è collegato a P2. A causa di questa differenza, il programma esempio deve avere due comandi aggiornati perché funzioni. Il comando era `HIGH 7`, dal momento che il circuito

del Potenziometro era collegato a P7, è ora **`HIGH 2`** visto che il circuito della fotoresistenza è collegato a P2. Per la stessa ragione, il comando che era **`RCTIME 7, 1, time`** è ora **`RCTIME 2, 1, time`**.

Programma di Esempio: TestFotoresistenza.bs2

Invece di ruotare la manopola del Potenziometro, il circuito viene collaudato esponendo la superficie sensibile alla luce della fotoresistenza a differenti intensità di luce. Quando il programma esempio sta girando, il Terminale di Debug dovrebbe visualizzare piccoli valori per alte luminosità ed alti valori per condizioni di bassa luminosità.

- √ Caricate e Lanciate TestFotoresistenza.bs2.
- √ Guardando il valore della variabile **`time`** sul Terminale di Debug annotate il valore con una illuminazione normale.
- √ Spengete la luce della stanza o create un'ombra con la vostra mano sul circuito e controllate di nuovo la variabile **`time`**. Dovrebbe essere un numero significativamente più grande.
- √ Se orientate la superficie sensibile della fotoresistenza verso la luce solare, la variabile **`time`** dovrebbe essere abbastanza piccola.

```
' Che cosa è un Microcontrollore- TestFotoresistenza.bs2
' Legge la fotoresistenza in un circuito RC-time usando il comando RCTIME.

' {$STAMP BS2}
' {$PBASIC 2.5}

time           VAR     Word

DO

  HIGH 2
  PAUSE 100
  RCTIME 2, 1, time
  DEBUG HOME, "time =  ", DEC5 time

LOOP
```

Il Vostro Turno – Usare un condensatore diverso per diverse condizioni di luce.

Sostituire il condensatore da 0.01 μF con un condensatore da 0.1 μF può essere utile per stanze più luminose o all'aperto. La misura di tempo con il condensatore da 0.1 μF sarà dieci volte più lunga, il che significa che il valore della variabile **`time`** visualizzato dal Terminale di Debug deve essere dieci volte più grande.

- √ Modificate il circuito sostituendo il condensatore da 0.01 µF con il condensatore da 0.1 µF.
- √ Rilanciate TestFotoresistenza.bs2 e verificate che le misure di RC-time siano all'incirca dieci volte il loro vecchio valore.
- √ Prima di andare al prossimo esercizio ritornate al circuito originale mostrato in Figura 7-2 rimuovendo il condensatore da 0.1 µF e sostituitelo con il condensatore da 0.01 µF.

ESERCIZIO #2: TRACCIARE IL GRAFICO DELLE MISURE DI LUCE

Le industrie spesso devono tenere sotto controllo molte informazioni da sensori per assicurarsi che le cose vadano nel giusto senso. Dal livello di luce nelle serre ai livelli di fluidi in una raffineria di petrolio alle temperature di un reattore nucleare, le persone responsabili del controllo di questi livelli spesso si affidano ai grafici delle misure dei sensori per ottenere le informazioni di cui abbisognano.

Conosciamo il programma Stamp Plot Lite

La Figura 7-4 mostra il software Stamp Plot Lite che graficizza le misure RC-time inviate dal BASIC Stamp. La linea illustrata in Figura 7-4 è molto più facile da comprendere delle 250 misure RC-time che sono state usate per tracciare quella linea. Vedendo il grafico da sinistra a destra, le misure RC-time diventano gradualmente più grandi ed improvvisamente diminuiscono. Dal momento che le misure RC-time aumentano quando la luce diminuisce e diminuiscono quando la luce aumenta, il grafico ci racconta una storia. Sembra quasi che il livello luminoso misurato diminuisca gradualmente, e quindi improvvisamente aumenti di nuovo.

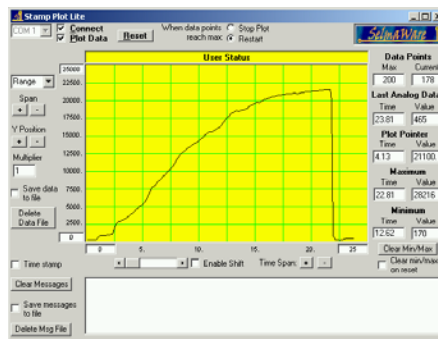



Figura 7-4
Grafico sul programma Stamp Plot Lite della misura del livello luminoso.

Questo grafico può raffigurare qualsiasi numero di scenari ai tecnici che lo leggono. Forse un microcontrollore in una serra accende le luci artificiali dopo che la luce solare sia scesa sotto un determinato livello. Forse un sistema di motori ed ingranaggi che mantiene puntato un pannello solare per la massima esposizione, è stato appena regolato dopo aver rilevato una diminuzione dell'esposizione luminosa. Indifferentemente dallo scenario, se avete familiarità con le misure graficizzate, utensili come lo Stamp Plot Lite possono veramente dare un senso ai dati. Esso lo fa convertendo elenchi di misure in grafici.

Stamp Plot Lite è gratuito per usi educativi grazie alla cortesia della SelmaWare Solutions e può essere installato dal CD Parallax o scaricato dal sito web Parallax o direttamente da http://www.selmaware.com/.

Scaricare ed installare Stamp Plot Lite

Prima di installare Stamp Plot Lite, dovete avere WinZip installato sul vostro PC o laptop. WinZip può essere installato dal CD della Parallax o scaricato dal sito www.winzip.com. Di seguito ci sono le istruzioni per scaricare ed installare Stamp Plot Lite dal sito web della Parallax:

√ Andate all'area Downloads del sito www.parallax.com.
√ Selezionate BASIC Stamp Software.
√ Scaricate il file etichettato "**Stamp Plot Lite** graphing utility…"
√ Salvate sul vostro disco StampPlot.zip.
√ Aprite StampPlot.zip clickando doppio.
√ Se state usando WinZip wizard, seguite i consigli sullo schermo, ed il programma di installazione si avvierà. Se state usando WinZip classic, dovrete clickare due volte sul file Setup.exe per iniziare l'installazione.
√ Per installare il software seguite i consigli del programma di installazione di Stamp Plot Lite.

Programmazione per inviare misure allo Stamp Plot Lite

L'invio dei valori da rappresentare allo Stamp Plot Lite è quasi lo stesso che inviare numeri al Terminale di Debug, ma ci sono alcune regole. Primo, i valori sono inviati usando solamente il formattatore **`DEC`** ed il carattere di controllo **`CR`**. Noi vogliamo rappresentare il valore della variabile **`time`**, così il valore decimale seguito dal ritorno carrello è tutto quello che dobbiamo inviare al Terminale di Debug.

```
DEBUG DEC time, CR
```

Potete anche inviare le impostazioni di visualizzazione allo Stamp Plot Lite inviando messaggi speciali fra virgolette. Questi messaggi sono chiamati codici di controllo. I codici di controllo vengono inviati allo Stamp Plot Lite all'inizio di un programma PBASIC. Sebbene possiate clickare e regolare tutte le impostazioni sul software stesso, è di norma più facile programmare il BASIC Stamp per inviare queste impostazioni allo Stamp Plot Lite al posto vostro. Segue un esempio di alcune impostazioni di configurazione per il prossimo programma esempio che faciliterà la lettura delle misure RC-time senza bisogno di alcuna regolazione delle impostazioni di visualizzazione dello Stamp Plot Lite.

```
DEBUG "!AMAX 1250", CR,
      "!TMAX 25", CR,
      "!TMIN 0", CR,
      "!SHFT ON", CR,
      "!RSET",CR
```

Per informazioni aggiuntive su come inviare valori e codici di controllo allo Stamp Plot Lite, lanciate il file di aiuto dello Stamp Pot Lite. Clickate Start, select programs, select Stamp Plot, quindi click Stamp Plot Help.

7

Programma di Esempio: PlotFotoresistenza.bs2

Seguite questi passi per rappresentare i dati della fotoresistenza:

√ Usate l'Editor del BASIC Stamp per Caricare e Lanciare PlotFotoresistenza.bs2.
√ Verificate che una sola colonna di valori scorra sul Terminale di Debug. La Figura 7-5 mostra un esempio.

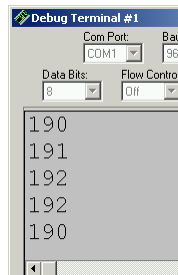



Figura 7-5
Esempio di scorrimento dei valori sul Terminale di Debug.

√ Annotate il numero della porta COM nel campo COM Port nell'angolo superiore sinistro del Terminale di Debug.

√ Usate il Menu Start di Windows per lanciare Stamp Plot Lite. Clickate *Start*, quindi selezionate *Programs* → *Stamp Plot* → *Stamp Plot Lite*.

√ Impostate il campo COM Port in Stamp Plot Lite allo stesso valore. La Figura 7-6 mostra un esempio dove il valore è COM1 nel Terminale di Debug, così com'è impostato Stamp Plot a COM1. Il vostro valore di COM port può essere un numero diverso. Semplicemente controllate quale valore è presente nel Terminale di Debug, quindi impostate lo stesso numero in Stamp Plot Lite.

√ Chiudete il Terminale di Debug (clickate il tasto *X* in alto a destra oppure clickate il tasto *Close* vicino al fondo della finestra).

√ Nella finestra di Stamp Plot Lite, clickate *Connect*, quindi clickate *Plot Data*. Dei segni di spunta devono apparire in ciascuna casella dopo averci clickato.

Figura 7-6
Impostazione della COM Port

Terminale di Debug (sinistra) e Stamp Plot Lite (destra).

√ Premete e rilasciate il tasto Reset sulla vostra Board of Education o HomeWork Board. Questo farà ripartire dall'inizio il programma BASIC Stamp, che invierà i comandi **DEBUG** che configureranno Stamp Plot Lite.

√ I dati verranno rappresentati appena clicckerete Plot Data. Tenete la vostra mano sulla fotoresistenza a diverse distanze per simulare diverse condizioni di illuminazione. Ricordate, più farete scura l'ombra, più alto sarà il valore nel grafico; più splendente la luce, minore il valore.

IMPORTANTE: Solamente un programma alla volta può usare una porta COM.

Prima di provare a lanciare un programma diverso usando l'Editor del BASIC Stamp, dovete deselezionare in Stamp Plot Lite le caselle *Connect* e *Plot Data*.

Prima di ricollegare Stamp Plot Lite (clickando le caselle Connect e Plot Data), dovete chiudere il Terminale di Debug.

```
' Che cosa è un Microcontrollore- PlotFotoresistenza.bs2
' Rappresenta graficamente i livelli di luce usando Stamp Plot Lite.
```

```
' {$STAMP BS2}
' {$PBASIC 2.5}

time           VAR     Word

DEBUG "!AMAX 1250", CR,
      "!TMAX 25", CR,
      "!TMIN 0", CR,
      "!SHFT ON", CR,
      "!RSET",CR
DO

  HIGH 2
  PAUSE 100
  RCTIME 2, 1, time
  DEBUG DEC time, CR

LOOP
```

7

Il Vostro Turno – Regolazione del Display

Span eTime Span hanno tasti + e – che potete clickare per aumentare o diminuire le scale orizzontali e verticali. Span è alla sinistra dell'area che visualizza il grafico, e potete usarlo per regolare i valori massimo e minimo rappresentati sul grafico. Time Span è sotto al grafico, e lo potete usare per cambiare la quantità di secondi rappresentati nella finestra.

- √ Sperimentate aumentando e diminuendo questi valori ed annotate i loro effetti sulla visualizzazione del grafico.

Se avete difficoltà nel trovare il vostro grafico, potete sempre premere e rilasciare il tasto Reset sulla vostra Board of Education o BASIC Stamp HomeWork Board per rimettere le impostazioni di partenza.

ESERCIZIO #3: TRACCIARE EVENTI LUMINOSI

Una delle più utili caratteristiche della memoria programma del BASIC Stamp è che togliete l'alimentazione, ma il programma non si perde. Appena lo riaccendete, il programma ricomincia da capo. Ciascuna porzione della memoria di programma che non viene usata per i programma, può essere usata per memorizzare dati. Questa memoria è specialmente utile per memorizzare dati che non volete siano dimenticati dal BASIC Stamp, perfino quando l'alimentazione viene tolta e ricollegata.

L'integrato che sul BASIC Stamp memorizza i programmi ed i dati viene mostrato in Figura 7-7. Questo integrato si chiama EEPROM. EEPROM sta per electrically erasable programmable read-only memory (memoria a sola lettura programmabile e cancellabile elettricamente). È una parola abbastanza lunga, e la pronuncia delle singole iniziali in EEPROM non è molto meglio. Quando si parla delle EEPROM, vengono di solito pronunciate "E-E-Prom".

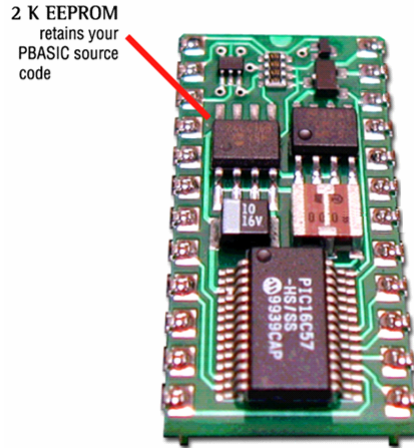



Figura 7-7
Integrato EEPROM sul BASIC Stamp

La Figura 7-8 mostra una finestra chiamata mappa della memoria. Potete vedere questa finestra clickando *Run* e selezionando *Memory Map*. La mappa della memoria usa colori diversi per mostrare come ambedue la RAM (variabili) e la EEPROM (memoria di programma) del BASIC Stamp vengano usate. La *Condensed EEPROM Map* in basso mostra che solamente una piccola frazione della memoria di programma viene usata per memorizzare il programma della fotoresistenza dall'Esercizio #2. Scorrendo verso il basso della *Detailed EEPROM Map* e contando i byte evidenziati in rosso, troverete che dei 2048 byte della EEPROM sono stati usati dal programma solamente 109. I restanti 1939 byte sono disponibili per la memorizzazione di dati.

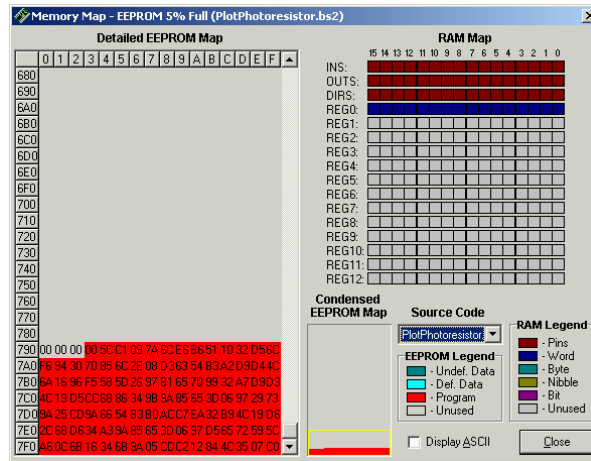



Figura 7-8
Mappa della Memoria

Per vedere questa finestra, clickate Run, e selezionate Memory Map.

2048 byte = 2 KB. Potete usare 2048 byte per memorizzare 2048 numeri diversi, ciascuno dei quali può rappresentare valori tra 0 e 255.

Il carattere maiuscolo 'B' sta per byte. Un carattere minuscolo 'b' sta per bit. Questo può fare una grande differenza perché 2048 Kb significa 2048 numeri differenti, ma ciascun numero è limitato ad un valore di 0 o di 1.

Sebbene in ambedue i casi la 'K' maiuscola e la 'k' minuscola vengano chiamati kilo, essi sono leggermente differenti. La 'K' maiuscola viene usata per indicare un kilobyte binario, che è $1 \times 2^{10} = 1024$. Quando ci si riferisce esattamente a 2000 byte, potete usare la k minuscola, che sta per kilo ($1 \times 10^3 = 1000$) nel sistema metrico.

Usare la EEPROM per la memorizzazione dei dati può essere molto utile per applicazioni remote. Un esempio di applicazione remota potrebbe essere un controllo di temperatura installato in un camion che trasporta cibo congelato. Un secondo esempio è una stazione di controllo del tempo meteorologico. Uno delle componenti della stazione meteo può memorizzare livelli luminosi per un uso futuro.

Dal momento che stiamo usando una fotoresistenza per misurare livelli luminosi, questo Esercizio introduce una tecnica per memorizzare e recuperare nella EEPROM livelli luminosi. In questo Esercizio, lancerete un programma esempio PBASIC che memorizza una serie di misure di luce nella EEPROM del BASIC Stamp. Dopo che il programma ha terminato, lancerete un secondo programma che recupera i valori dalla EEPROM e li visualizza nel Terminale di Debug.

Programmazione della Memorizzazione dei Dati a Lungo Termine

Viene usato il comando **WRITE** per memorizzare i valori nella EEPROM, e per recuperare quei valori il comando **READ**.

La sintassi per il comando **WRITE** è:

WRITE ***Location, {WORD} Data Item***

Per esempio, se volete scrivere il valore 195 all'indirizzo 7 nella EEPROM, potreste usare il comando:

```
WRITE 7, 195
```

I valori Word possono andare da 0 a 65535 mentre i valori byte possono contenere solamente numeri da 0 a 255. Un valore word occupa lo spazio di due byte. Se volete scrivere un valore word nella EEPROM, dovrete usare il parametro opzionale ***Word***. Siate prudenti. Dal momento che una word occupa due byte, dovrete scartare uno degli indirizzi in formato byte nella EEPROM prima che possiate scrivere un altra word. Diciamo che dobbiate salvare due valori word nella EEPROM: 659 e 50012. Potreste iniziare dall'indirizzo 8, ma dovrete scrivere il secondo valore all'indirizzo 10.

```
WRITE 8, Word 659
WRITE 10, Word 50012
```

È possibile sovrascrivere il programma? Si, e se lo fate, è probabile che il programma possa comportarsi stranamente o fermarsi del tutto. Dal momento che le istruzioni del programma PBASIC risiedono nella parte alta degli indirizzi della EEPROM, la cosa migliore è di usare gli indirizzi più bassi per memorizzare numeri con il comando **WRITE**.

Come faccio a sapere se l'indirizzo che sto usando è troppo grande? Potete usare la mappa della memoria per trovare il massimo valore non usato dal programma PBASIC. I numeri sulla sinistra della mappa di memoria dettagliata mostrata nella Figura 7-8 a pagina 203 sono numeri esadecimali, e 792 è il valore più alto non occupato da una istruzione di programma. Esadecimale significa a base-16, così 792 in realtà è:

$7\times16^2 + 9\times16^1 + 2\times16^0$

$= 7\times256 + 9\times16 + 2\times1$

$= 1938.$

Potete anche programmare il BASIC Stamp per fare questa conversione al posto vostro con l'uso del comando DEBUG il formattatore **DEC** e l'operatore esadecimale **$** come questo:

```
DEBUG DEC $792
```

Programma di Esempio: StoreLightMeasurementsInEeprom.bs2

Questo programma esempio mostra come usare il comando **WRITE** facendo misure di luce ogni 5 secondi per 2 ½ minuti e memorizzandole nella EEPROM.

- √ Caricate e Lanciate StoreLightMeasurementsToEeprom.bs2 .
- √ Registrate le misure visualizzate sul Terminale di Debug cosi che possiate verificare le misure rilette dalla EEPROM.

Aumentate gradualmente l'ombra sulla fotoresistenza durante il periodo di prova di 2 ½ minuti per avere dati significativi.

```
' Che cosa è un Microcontrollore- StoreLightMeasurementsInEeprom.bs2
' Scrive le misure di luce nella EEPROM.

' {$STAMP BS2}
' {$PBASIC 2.5}

time           VAR     Word
eepromAddress  VAR     Byte

DEBUG "Inizio della misurazione...", CR, CR,
      "Misurazione   Valore", CR,
      "-----------   ------", CR

PAUSE 1000

FOR eepromAddress = 0 TO 58 STEP 2
  HIGH 2
  PAUSE 5000
  RCTIME 2, 1, time
  DEBUG DEC2 eepromAddress,
        "             ", DEC time, CR
  WRITE eepromAddress, Word time
NEXT

DEBUG "Fatto. Ora, lanciate:", CR,
      "ReadLightMeasurementsFromEeprom.bs2"
END
```

7

Come funziona StoreLightMeasurementsInEeprom.bs2

Il ciclo **FOR…NEXT** che misura i valori RC-time e li memorizza nella EEPROM deve essere contato a passi di due perché nella EEPROM vengono scritti valori word.

```
FOR eepromAddress = 0 to 58 STEP 2
```

Il comando **RCTIME** carica le misure del tempo nella variabile **time** in formato word.

```
    RCTIME 2, 1, time
```

La variabile **time** viene memorizzata all'indirizzo dato dal valore corrente della variabile **eepromAddress** ad ogni iterazione del ciclo. Ricordate, la variabile **eepromAddress** viene incrementata di due ad ogni iterazione del ciclo perché una variabile **Word** occupa due byte. L'indirizzo per un comando **WRITE** è sempre espresso in termine di byte.

```
    WRITE eepromAddress, Word time
  NEXT
```

Programmazione del recupero dei Dati

Per recuperare questi valori dalla EEPROM, potete usare il comando **READ**. La sintassi del comando **READ** è:

READ *Location, {WORD} Data Item*

Mentre il comando **WRITE** può usare sia le costanti che le variabili, l'argomento **DataItem** del comando **READ** deve essere una variabile, perché deve memorizzare il valore estratto dalla EEPROM dal comando **READ**.

Assumiamo che la variabile **eepromValueA** ed **eepromValueB** siano variabili **Word**, e **littleEE** sia una variabile. Seguono alcuni comandi per recuperare i valori che avete memorizzato usando il comando write.

```
READ 7, littleEE
READ 8, Word eepromValueA
READ 10, Word eepromValueB
```

Programma di Esempio: ReadLightMeasurementsFromEeprom.bs2

Questo programma esempio dimostra come usare il comando **READ** per recuperare le misure di intensità di luce che erano state memorizzate nella EEPROM dal programma StoreLightMeasurementsInEeprom.bs2.

√ Dopo aver completato StoreLightMeasurementsToEeprom.bs2, scollegate e ricollegate l'alimentazione del BASIC Stamp per provare che i dati non si sono cancellati dalla EEPROM del BASIC Stamp quando avete scollegato l'alimentazione. Chiudete e riaprite anche l'Editor del BASIC Stamp.

- √ Lasciate spento il BASIC Stamp fino a che siete pronti a lanciare ReadLightMeasurementsFromEeprom.bs2; altrimenti, comincerà di nuovo ad effettuare misure.
- √ Caricate e Lanciate ReadLightMeasurementsFromEeprom.bs2.

Confrontate la tabella che viene visualizzata da questo programma con quella visualizzata da StoreLightMeasurementsInEeprom.bs2, e verificate che i valori siano gli stessi.

```
' Che cosa è un Microcontrollore- ReadLightMeasurementsFromEeprom.bs2
' Legge le misure di luminosità dalla EEPROM.

' {$STAMP BS2}
' {$PBASIC 2.5}

time           VAR     Word
eepromAddress  VAR     Byte

DEBUG "Recupero delle Misure", CR, CR,
      "Misure   Valori", CR,
      "-----------   ------", CR

FOR eepromAddress = 0 TO 58 STEP 2
  READ eepromAddress, Word time
  DEBUG DEC2 eepromAddress, "           ", DEC time, CR
NEXT

END
```


Come funziona ReadLightMeasurementsFromEeprom.bs2

Come con il comando **`WRITE`**, il comando **`READ`** usa indirizzi in formato byte. Dal momento che nella EEPROM sono lette word, ad ogni iterazione del ciclo **`FOR…NEXT`** deve essere aggiunto 2 alla variabile **`eepromAddress`**.

```
FOR eepromAddress = 0 to 58 STEP 2
```

Il comando **`READ`** prende il valore di formato word dall'indirizzo **`eepromAddress`**. Questo valore viene caricato nella variabile **`time`**.

```
  READ eepromAddress, Word time
```

Il valore delle variabili **`time`** ed **`eepromAddress`** vengono visualizzate nelle colonne nella tabella del Terminale di Debug.

```
    DEBUG DEC2 eepromAddress, "           ", DEC time, CR
NEXT
```

Il Vostro Turno – Tracciare i Dati Memorizzati

√ Modificate ReadLightMeasurementsFromEeprom.bs2 cosi che visualizzi i dati con Stamp Plot Lite. Ricordate, l'istruzione **DEBUG** deve visualizzare solamente il valore ed il ritorno carrello.

ESERCIZIO #4: UN SEMPLICE MISURATORE DI LUCE

L'informazione del sensore di luce possono essere comunicate in svariati modi. Il misuratore di luce con cui lavorerete in questo Capitolo cambia il ritmo con cui il display lampeggia secondo l'intensità di luce che rileva.

Componenti del Misuratore di Luce

(1) Fotoresistenza
(1) Resistenza – 220 Ω (rosso-rosso-marrone)
(1) Condensatore – 0.01 µF
(1) Condensatore – 0.1 µF
(1) Display LED a 7-segmenti
(8) Resistenze – 1 kΩ (marrone-nero-rosso)
(6) Ponticelli

Costruzione del Misuratore di Luce

La Figura 7-9 mostra gli schemi del circuito del display LED a 7-segmenti e della fotoresistenza che saranno usati per costruire il misuratore di luce, e la Figura 7-10 mostra lo schema di cablaggio del circuito. Il circuito della fotoresistenza è lo stesso che avete usato negli ultimi due esercizi, ed il circuito del display LED a 7-segmenti è lo stesso che era controllato dal BASIC Stamp nel Capitolo #6.

√ Costruite il circuito mostrato in Figura 7-9 ed in Figura 7-10.
√ Collaudate il LED a 7-segmenti per assicurarvi che sia collegato correttamente usando SegmentTestWithHighLow.bs2 dal Capitolo #6, Esercizio #2.

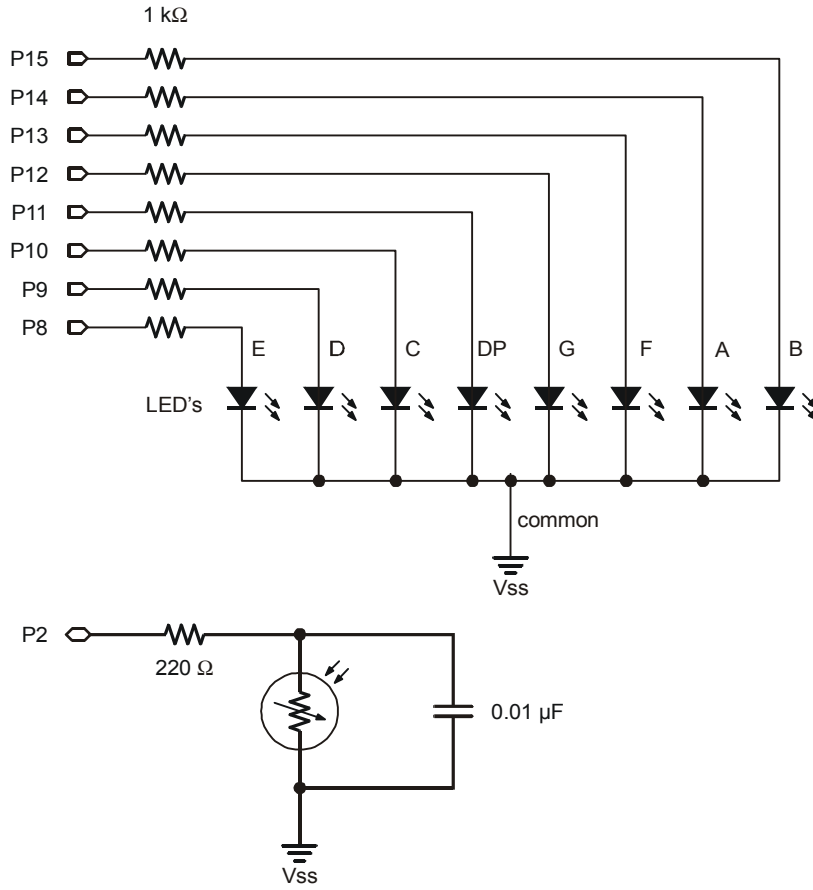



Figura 7-9
Schema Elettrico del Misuratore di Luce.

7

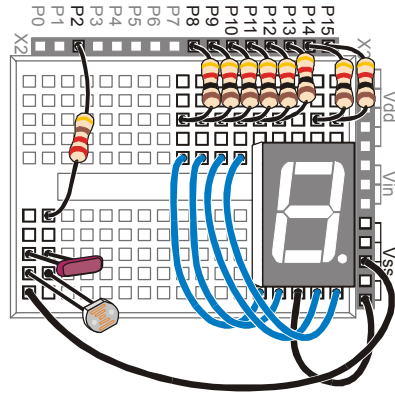



Figura 7-10
Schema di Cablaggio per la Figura 7-9

L'uso delle Subroutine

La maggioranza dei programmi che avete scritto fino ad ora, operano all'interno di un ciclo **DO…LOOP**. Dal momento che tutti gli Esercizi principali operano all'interno di un ciclo **DO…LOOP**, vengono di solito chiamate routine principali. Quando aggiungete altri circuiti ed altre utili funzioni al vostro programma, può divenire difficoltoso tenere traccia del codice nella routine principale. I vostri programmi sarebbero molto più facili da leggere se poteste organizzarli in segmenti più piccoli che assolvano compiti specifici. Il PBASIC possiede alcuni comandi che potete usare per far saltare il programma fuori dalla routine principale, effettuare un compito, e ritornare nello stesso punto della routine principale da cui era partita. Questo vi permetterà di tenere ciascun segmento che esegue un compito particolare in qualche parte diversa dalla vostra routine principale. Ogni volta che necessitate di eseguire uno di questi compiti, potete scrivere all'interno della routine principale un comando che dice al programma di saltare al quel compito, eseguirlo, e quando finito ritornare al punto da cui era partito. Questo procedimento, si chiama esecuzione di una subroutine.

La Figura 7-11 mostra un esempio di subroutine e di come viene usata. Il comando **GOSUB Subroutine_Name** forza il programma a saltare all'etichetta **Subroutine_Name:**. Quando il programma arriva all'etichetta, prosegue eseguendo i comandi presenti fino a che giunge ad un'istruzione **RETURN**. Quindi, il programma torna indietro all'istruzione successiva al comando **GOSUB**. Nel caso dell'esempio di Figura 7-11, l'istruzione successiva è **DEBUG "Next command"**.

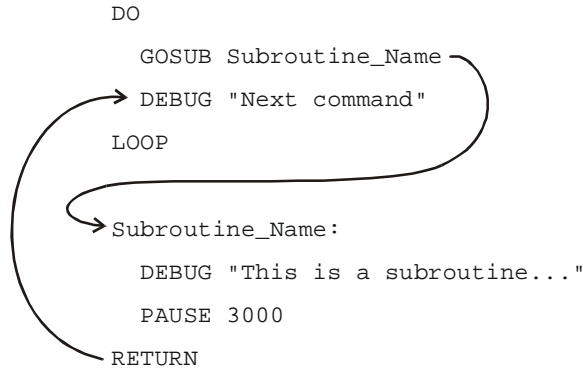



Figura 7-11
Come lavorano
le Subroutine

Che cosa è un Etichetta? Un etichetta è un nome che può essere usato come segnaposto nel vostro programma. **GOSUB** è uno dei comandi che potete usare per saltare ad un etichetta. Alcuni altri sono **GOTO**, **ON GOTO**, ed **ON GOSUB**. Potete usare questi comandi per saltare alle etichette. Un'Etichetta deve terminare con due punti (**:**), e per essere corretti, separate le parole con il carattere sottolineato(**_**). Quando scegliete un nome per un'etichetta, assicuratevi di non sceglier un nome riservato. Le altre regole per il nome di etichetta sono le stesse per nominare le variabili e sono elencate nel riquadro informativo a pagina 56.

Programma di Esempio: SimpleSubroutines.bs2

Questo programma esempio mostra come funzionano le subroutine inviando messaggi al Terminale di Debug.

- √ Esaminate SimpleSubroutines.bs2 e cercate di indovinare l'ordine in cui verranno eseguiti i comandi **DEBUG**.
- √ Digitate e lanciate il programma.
- √ Confrontate il comportamento attuale del programma con le vostre previsioni.

```
' Che cosa è un Microcontrollore- SimpleSubroutines.bs2
' Dimostrazione di come funzionano le subroutine.

' {$STAMP BS2}
' {$PBASIC 2.5}

DO
```

```
  DEBUG CLS, "Inizio della routine principale.", CR
  PAUSE 2000
  GOSUB First_Subroutine
  DEBUG "Indietro alla routine principale.", CR
  PAUSE 2000
  GOSUB Second_Subroutine
  DEBUG "Ripetizione della routine principale...", CR
  PAUSE 2000

LOOP


First_Subroutine:

  DEBUG "  Esecuzione della prima "
  DEBUG "subroutine.", CR
  PAUSE 3000

RETURN


Second_Subroutine:

  DEBUG "  Esecuzione della seconda "
  DEBUG "subroutine.", CR
  PAUSE 3000

RETURN
```

Come funziona SimpleSubroutines.bs2

La Figura 7-12 mostra come funziona nella routine principale la chiamata della **`First_Subroutine`** (il ciclo **`DO…LOOP`**). Il comando **`GOSUB First_Subroutine`** invia il programma all'etichetta **`First_Subroutine:`**. Vengono eseguiti i tre comandi all'interno di quella subroutine. Quando il programma arriva all'istruzione **`RETURN`**, esso salta indietro al comando che viene subito dopo il comando **`GOSUB`**, che è **`DEBUG`** " Indietro alla routine principale.", **`CR`**.

Che cosa è una chiamata ad una subroutine? Quando usate il comando **`GOSUB`** per far saltare il programma ad una subroutine, viene detto chiamata ad una subroutine.

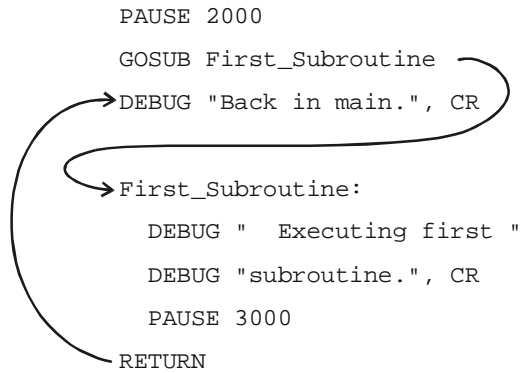



Figura 7-12
Prima chiamata di una Subroutine

La Figura 7-13 mostra un secondo esempio dello stesso procedimento con la seconda chiamata di una subroutine (**`GOSUB Second_Subroutine`**).

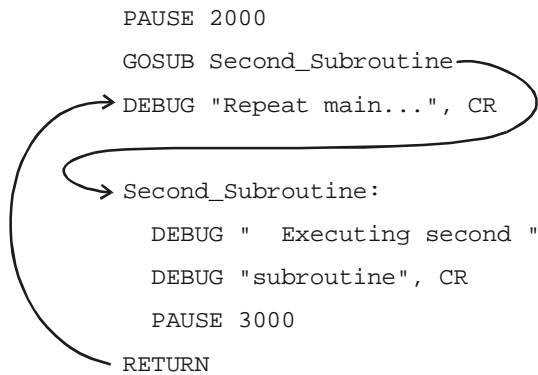



Figura 7-13
Seconda chiamata di una Subroutine

Il Vostro Turno – Aggiunta e Nidificazione delle Subroutine

Potete aggiungere subroutine alle due che sono già nel programma e chiamarle dall'interno della routine principale.

√ Aggiungete a SimpleSubroutines.bs2 l'esempio di subroutine della Figura 7-11 a pagina 211.

√ Fate tutti i necessari aggiustamenti ai comandi **DEBUG** in modo che la visualizzazione sia corretta con tutte e tre le subroutine.

Potete anche chiamare una subroutine dall'interno di un'altra. Questo viene chiamato nidificazione di subroutine o subroutine annidate.

√ Provate a spostare il comando **GOSUB** che chiama **Subroutine_Name** all'interno di una delle altre subroutine, e vedete come funziona.

Quando annidate le subroutine la regola è non più di quattro nidificazioni. Per maggiori dettagli vedere il Manuale del BASIC Stamp. Cercate **GOSUB** e **RETURN**.

Misuratore di Luce usando le Subroutine

I segmenti del display si accendono con un movimento circolare che diventa più veloce quando la luce che giunge sulla fotoresistenza aumenta. Quando la luce si affievolisce, il movimento circolare del display LED a 7-segmenti rallenta.

Il programma che gestisce il misuratore di luce dovrà eseguire tre diverse operazioni:

1. Leggere la fotoresistenza.
2. Calcolare quanto attendere prima di aggiornare il display LED a 7-segmenti.
3. Aggiornare il display LED 7-segmenti.

Ciascuna operazione è contenuta all'interno della sua subroutine, e la routine principale **DO…LOOP** ciclicamente chiamerà ciascuna subroutine in sequenza, ricominciando da capo.

Programma di Esempio: LightMeter.bs2

La Condizione di Luce Controllata fa una grande differenza. Per i migliori risultati, eseguite questa prova in una stanza illuminata da luce fluorescente senza luce solare diretta (chiudete le serrande). Per informazioni su come calibrare questo misuratore per altre condizioni di illuminazione, Vedere la sezione Il Vostro Turno.

√ Caricate e Lanciate LightMeter.bs2.

√ Verificate che il movimento circolare visualizzato dal display LED a 7-segmenti venga controllato dalle condizioni di illuminazione che la fotoresistenza sta

rilevando. Fate le prove proiettando un ombra sulla fotoresistenza con la vostra mano o con un foglio di carta.

```
' Che cosa è un Microcontrollore- LightMeter.bs2
' Indica il livelli luminoso usando un display a 7-segmenti.

' {$STAMP BS2}
' {$PBASIC 2.5}

index          VAR     Nib              ' Dichiarazione delle Variabili.
time           VAR     Word

OUTH = %00000000               ' Inizializzazione del display 7-segmenti.
DIRH = %11111111

DO                                      ' Routine Principale.

  GOSUB Get_Rc_Time
  GOSUB Delay
  GOSUB Update_Display

LOOP

                                        ' Subroutine

Get_Rc_Time:                            ' Subroutine RC-time

  HIGH 2
  PAUSE 3
  RCTIME 2, 1, time

RETURN


Delay:                                  ' Subroutine di ritardo.

  PAUSE time

RETURN


Update_Display:                ' Subroutine di aggiornamento del Display.

  IF index = 6 THEN index = 0
  '                BAFG.CDE
  LOOKUP index, [ %01000000,
                  %10000000,
                  %00000100,
                  %00000010,
                  %00000001,
```

7

```
                 %00100000 ], OUTH
  index = index + 1

RETURN
```

Come funziona LightMeter.bs2

Le prime due linee del programma dichiarano le variabili. Non è importante se queste variabili vengono usate nelle subroutine o nella routine principale, la miglior cosa è sempre dichiarare le variabili (e le costanti) all'inizio del vostro programma. Dal momento che questa è una pratica comune, questa sezione del codice ha un nome, Dichiarazione delle Variabili. Questo nome viene mostrato nel commento alla destra della prima dichiarazione di variabile.

```
index VAR Nib                  ' Dichiarazione delle Variabili.
time  VAR Word
```

Molti programmi hanno anche cose che devono essere eseguite solamente una volta all'inizio del programma. Un esempio è l'impostazione bassa di tutti i pin I/O dei 7-segmenti. Anche questa sezione del programma PBASIC ha un nome, Inizializzazione.

```
OUTH = %00000000     ' Inizializzazione del display 7-segmenti.
DIRH = %11111111
```

Il prossimo segmento di codice è la routine principale. La routine principale per prima cosa chiama la subroutine **`Get_Rc_Time`**. Quindi, chiama la subroutine **`Delay`**, ed infine, chiama la subroutine **`Update_Display`**. Ricordatevi che il programma gira continuamente attraverso le tre subroutine alla massima velocità.

```
DO                             ' Routine principale.
  GOSUB Get_Rc_Time
  GOSUB Delay
  GOSUB Update_Display
LOOP
```

Tutte le subroutine sono di solito messe dopo la routine principale. Il nome della prima subroutine è **`Get_Rc_Time:`**, ed effettua la misura del RC-time del circuito della fotoresistenza. Questa subroutine ha un comando **`PAUSE`** che carica il condensatore. L'argomento ***`Duration`*** di questo comando è piccolo perché deve solamente attendere che il condensatore sia carico. Notate che il comando **`RCTIME`** imposta il valore della variabile **`time`**. Questa variabile sarà usata dalla seconda subroutine.

```
                                ' Subroutine
Get_Rc_Time:                   ' subroutine RC-time
  HIGH 2
  PAUSE 3
  RCTIME 2, 1, time
RETURN
```

Il nome della seconda subroutine è **`Delay`**, e tutto quello che contiene è un comando **`PAUSE`**. Se volete effettuare calcoli extra sul valore della variabile **`time`** prima di usarla nel comando **`PAUSE`**, sarebbe corretto eseguirli in questa subroutine.

```
Delay:
  PAUSE time
RETURN
```

La terza subroutine viene chiamata **`Update_Display`**. Il comando **`LOOKUP`** in questa subroutine contiene una tabella con sei gruppi di bit che vengono usati per creare il movimento circolare dei segmenti esterni del display. Aggiungendo 1 alla variabile **`index`** ogni volta che la subroutine viene chiamata, viene posto nel **`OUTH`** il gruppo successivo dei bit per l'accensione del display. Nella tabella del comando **`LOOKUP`** ci sono valori dell'**`index`** che vanno da 0 a 5. che cosa accade quando il valore di **`index`** diventa 6? Il comando lookup non sa andare automaticamente indietro al primo valore, ma per risolvere questo problema potete usare una istruzione **`IF…THEN`**. Il comando **`IF index = 6 THEN index = 0`** reimposta a 0 il valore di **`index`** ogni volta che giunge a 6. Inoltre forza la sequenza dei gruppi di bit messa in **`OUTH`** a ripetersi di continuo. Questo a sua volta, fa ripetere il movimento circolare dei segmenti del display LED a 7-segmenti.

```
Update_Display:
  IF index = 6 THEN index = 0
  '                BAFG.CDE
  LOOKUP index, [ %01000000,
                  %10000000,
                  %00000100,
                  %00000010,
                  %00000001,
                  %00100000 ], OUTH
  index = index + 1
RETURN
```

Il Vostro Turno – Regolazione dell'Hardware e del Software del Misuratore.

Ci sono due modi per cambiare la sensibilità del misuratore. Per prima cosa può essere cambiato il software. Per esempio, se voi moltiplicate per 10 la variabile time della

subroutine **`Delay`** , il display LED a 7-segmenti ruoterà ad un decimo della velocità, e se dividete la variabile time per 2, ruoterà al doppio della velocità.

√ Modificate LightMeter.bs2 in modo che la variabile **`time`** sia moltiplicata per 10. Il modo più facile è di effettuare il seguente cambiamento da:

```
PAUSE time
```

a

```
PAUSE time * 10
```

Nella subroutine **`Delay`**.

√ Lanciate il programma modificato e provatelo per assicurarvi che i segmenti del display LED a 7-segmenti ruotino ad un decimo della precedente velocità.
√ Potete anche provare a moltiplicare la variabile **`time`** per altri valori come per esempio 5 o 20, o dividerla per 2 usando **`PAUSE time / 2`**.

Potete anche far ruotare il display ad un decimo della velocità cambiando il condensatore da 0.01µF con il condensatore da 0.1 µF. Ricordate comunque che quando usate un condensatore dieci volte più grande, la misura del tempo RC-time sarà dieci volte più lunga.

√ Sostituite il condensatore da 0.01 µF con il condensatore da 0.1 µF.
√ Lanciate il programma e vedete se si verifica l'effetto voluto.

Che cosa è meglio, modificare il software o l'hardware? Dovreste sempre cercare di usare il meglio dei due mondi. Scegliete un condensatore che vi dia la misurazione più accurata nella gamma più ampia di livelli luminosi. Una volta che il vostro hardware è al meglio, usate il software per regolare automaticamente il misuratore di luce in maniera che sia adeguato all'utilizzo, per ambienti interni e per ambienti esterni. Questo produce una mole considerevole di prove ed aggiustamenti, ma fa parte del processo di sviluppo del progetto.

SOMMARIO

Questo Capitolo ha introdotto un secondo modo di usare il comando **RCTIME** usandolo per misurare i livelli luminosi tramite una fotoresistenza. Similmente al Potenziometro, la fotoresistenza è una Resistenza variabile. Diversamente dal Potenziometro, la resistenza della fotoresistenza cambia con il livello di luminosità invece che con la posizione. È stato usato Stamp Plot Lite per rappresentare graficamente misure di luminosità successive, e sono stati anche spiegati metodi per registrare ed interpretare dati graficamente. Sono stati usati i comandi **WRITE** e **READ** per memorizzare e recuperare valori nella EEPROM del BASIC Stamp. La EEPROM è stata quindi usata in una applicazione di raccolta dati della funzione RC-time. Nell'ultimo esercizio di questo Capitolo, è stata sviluppata un'applicazione per un misuratore di luminosità. Questa applicazione ha usato le subroutine per eseguire tre compiti diversi necessari per il funzionamento del misuratore di luminosità.

7

Domande

1. Quale genere di cose diverse possono rilevare i sensori?
2. Qual è il nome del composto chimico che rende la fotoresistenza sensibile alla luce?
3. In cosa è uguale una fotoresistenza ad un Potenziometro? Ed in cosa è diversa?
4. Quale genere di circuito è stato usato per misurare la fotoresistenza?
5. Quale genere di misurazione esegue il BASIC Stamp con questo circuito?
6. In cosa differiva il circuito di prova della fotoresistenza dal circuito di prova del Potenziometro del Chapter #5? Quali cambiamenti si sono dovuti fare al programma di prova della fotoresistenza per farlo funzionare con il nuovo circuito?
7. Se la linea della fotoresistenza tracciata in Stamp Plot Lite aumenta, che cosa ci dice circa il livello luminoso misurato?
8. Possono nello stesso tempo due programmi diversi usare una porta COM?
9. Qual è la differenza tra l'impostazione Span e Time Span in Stamp Plot Lite?
10. Che cosa significa la sigla EEPROM?
11. Quanti byte può memorizzare la EEPROM del BASIC Stamp? Quanti bit può memorizzare?
12. Quale comando usate per memorizzare un valore nella EEPROM? Quale comando usate per recuperare un valore dalla EEPROM? Quale dei due richiede una variabile?

13. Ci sono due differenze tra memorizzare byte e word usando le EEPROM. Quale sono?
14. Che cos'è un'etichetta?
15. Che cos'è una subroutine?
16. Quale comando si usa per chiamare una subroutine?
17. Quale comando si usa per terminare una subroutine?
18. Che cosa si usa per nominare una subroutine?
19. Quale differenza c'è fra la routine principale ed una subroutine?

Esercizi

1. Disegnate lo schema elettrico di un circuito RC-time con fotoresistenza collegato a P5.
2. Modificate TestFotoresistenza.bs2 in modo che lavori in un circuito collegato a P5 invece che a P2.
3. Modificate TestFotoresistenza.bs2 in modo che visualizzi le stesse misure del condensatore da 0.01 μF con il condensatore da 0. 1 μF.
4. Immaginate che nel vostro circuito RC-time stiate usando un condensatore più piccolo o che stiate dividendo la misurazione per un valore che mantenga il risultato al di sotto di 256. questo vi permetterà di memorizzare nella EEPROM il doppio dei valori. Spiegate perché.
5. Spiegate come modifichereste LightMeter.bs2 in modo che venga invertito il senso di rotazione dei segmenti del display LED a 7-segmenti.

Progetti

1. In un Capitolo precedente, avete usato un tasto per far lampeggiare un LED. Invece di usare un tasto, usate una fotoresistenza per far lampeggiare il LED quando proiettate un'ombra sulla fotoresistenza. Suggerimento: potete usare un'istruzione **`IF…THEN`** con gli operatori "maggiore di", "minore di" per decidere se la vostra misura di **`time`** sia sopra o sotto un determinato valore. Per "maggiore di" si usa l'operatore **`>`**, e per "minore di" si usa **`<`**.
2. Usate la vostra fotoresistenza ed un programma PBASIC per fare un semplice encoder. Un encoder è un dispositivo che rileva il passaggio dei raggi di una ruota. Usate la vostra fotoresistenza come sensore per rilevare il passaggio dei raggi ed usate le vostre mani al posto dei raggi. Potete emulare il passaggio dei raggi davanti al sensore, facendo passare la vostra mano ripetutamente davanti alla fotoresistenza. Usate il Terminale di Debug per visualizzare quante volte avete passato la vostra mano davanti alla fotoresistenza. Dopo alcune

calibrazioni, per emulare i raggi, potete anche passare il vostro dito davanti alla fotoresistenza.

3. Progettate un controllo di ombreggiamento automatizzato usando un Servomotore ed una fotoresistenza. Potete farlo attaccando un pezzo di cartoncino o di balsa alla crociera del vostro Servomotore. Quindi, montate il Servomotore sopra la fotoresistenza in modo che il cartoncino o la balsa, facciano ombra sulla fotoresistenza. Il Servomotore dovrebbe essere in grado di regolare il livello di ombreggiatura modificando la posizione della crociera del Servomotore. Scrivete un programma faccia muovere il Servomotore per aumentare l'ombreggiatura sulla fotoresistenza quando la luce è intensa e la faccia diminuire quando la luce si affievolisce.
4. Espandete il progetto precedente in modo che memorizzi il livello di illuminazione misurato ogni mezzo minuto per un periodo di 20 minuti, e quindi lo visualizzi con Stamp Plot Lite. Dopo che tutto questo funziona correttamente, modificate il programma in modo che alte luminosità siano visualizzate nel grafico da livelli alti e viceversa.

7

Ulteriori Approfondimenti

"*Applied Sensors*", Student Guide, Version 2.0, Parallax Inc., 2003

In questo testo sono trattati in maniera più approfondita, la misurazione della luce tramite fotodiodi, unità scientifiche e matematiche, e la raccolta di dati insieme con altre applicazioni dei sensori.

"*Industrial Control*", Student Guide, Version 2.0, Parallax Inc., 2002

Stamp Plot Lite è stato sviluppato insieme a questo testo per dimostrare i fondamenti delle tecniche usate nei processi di controllo industriali.

Capitolo #8: Frequenza e Suono

LA VOSTRA GIORNATA ED I BIP ELETTRONICI

Vengono forniti di seguito alcuni esempi di bip che potete udire durante una giornata normale: Il forno a microonde emette un bip quando ha finito di cuocere il vostro cibo. Il telefono cellulare emette toni diversi che somigliano a canzoni per attirare la vostra attenzione su una chiamata in arrivo. Il bancomat emette un bip per ricordarvi di riprendere la vostra tessera. Il registratore di cassa emette un bip per informare il cassiere che l'oggetto passato sopra il lettore di codice a barre è stato riconosciuto. Molte calcolatrici emettono un bip quando vengono premuti i tasti sbagliati. Non ci dimentichiamo che potreste aver iniziato la vostra giornata con il bip della sveglia digitale.

MICROCONTROLLORI, ALTOPARLANTI, BIP E SEGNALI ON/OFF

All'incirca tutti i bip elettronici che sentite nella routine della vostra giornata sono emessi da altoparlanti collegati a microcontrollori. Il microcontrollore crea questi bip inviando rapidi segnali on/off a vari tipi di altoparlanti. La velocità di questi segnali on/off viene chiamata frequenza, ed essa determina il tono o altezza del bip. Ogni volta che un segnale on/off si ripete, viene chiamato ciclo. Vedrete spesso il numero di cicli per secondo espresso come Hertz, e viene abbreviato in Hz. Per esempio, una delle frequenze più comuni del bip che aiuta le macchine ad attrarre la vostra attenzione è 2 kHz. Questo significa che il segnale on/off si ripete 2000 volte al secondo.

Conosciamo l'Altoparlante Piezoelettrico

In questo Esercizio, esperimenterete l'invio di una varietà di segnali ad un piccolo, comune, ed economico altoparlante chiamato altoparlante piezoelettrico. Il suo simbolo schematico ed il disegno del componente sono mostrati in Figura 8-1.

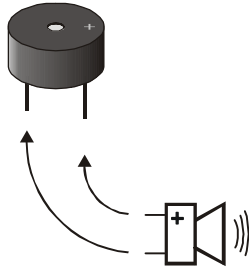



Figura 8-1
Altoparlante Piezoelettrico
Simbolo schematico e
disegno del componente

Un Altoparlante piezoelettrico è comunemente chiamato altoparlante piezo o cicalino piezo, viene anche chiamato con la parola inglese "Buzzer".

ESERCIZIO #1: COSTRUZIONE E COLLAUDO DELL'ALTOPARLANTE

In questo Esercizio, costruirete e collauderete il circuito dell'altoparlante piezoelettrico.

Componenti del Circuito dell'Altoparlante

(1) Altoparlante piezoelettrico
(2) Ponticelli

Costruzione del circuito dell'Altoparlante piezoelettrico

Il terminale negativo dell'altoparlante piezoelettrico dovrebbe essere collegato a Vss, ed il terminale positivo dovrebbe essere collegato ad un pin I/O. Si dovrà quindi programmare il BASIC Stamp per inviare segnali on/off al terminale positivo dell'altoparlante piezoelettrico.

√ Costruite il circuito mostrato in Figura 8-2.

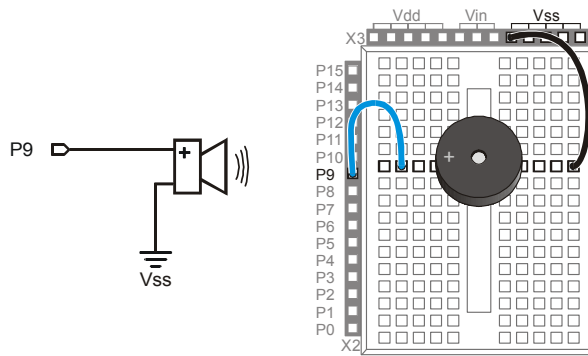



Figura 8-2
Circuito dell'Altoparlante piezoelettrico Schema Elettrico e Schema di Cablaggio

Come funziona il circuito dell'Altoparlante piezoelettrico

Quando la corda di una chitarra vibra, provoca cambiamenti nella pressione dell'aria. Questi cambiamenti nella pressione dell'aria sono quelli che voi rilevate come toni. Più veloce il cambiamento della pressione dell'aria, più acuto è il tono, e più lento il cambiamento della pressione dell'aria, più grave è il tono. L'elemento all'interno del contenitore plastico dell'altoparlante piezo si chiama elemento piezoelettrico. Quando dei segnali on/off vengono applicati al terminale positivo dell'altoparlante, l'elemento piezoelettrico vibra, causando quindi dei cambiamenti nella pressione dell'aria proprio come fa una corda di chitarra. Come con la corda di chitarra, il vostro orecchio percepisce i cambiamenti della pressione dell'aria causati dall'altoparlante piezoelettrico, che viene quindi percepito come un bip od un tono.

Programmazione del Controllo dell'Altoparlante

Il comando **`FREQOUT`** è un modo conveniente per inviare segnali ad un altoparlante per generare un suono. Il *Manuale del BASIC Stamp* mostra la sintassi del comando simile alla seguente:

FREQOUT *Pin, Duration, Freq1 {, Freq2}*

Come per molti degli altri comandi usati in questo libro, ***`Pin`*** è un valore che potete usare per scegliere quale pin I/O del BASIC Stamp I/O usare. L'argomento ***`Duration`*** è un valore che dice al comando **`FREQOUT`** quanto a lungo deve essere suonato il tono, in millisecondi. L'argomento ***`Freq1`*** viene usato per impostare la frequenza del tono, in Hertz. C'è un argomento opzionale ***`Freq2`*** che può essere usato per miscelare frequenze.

Viene qui indicato come inviare un tono al pin I/O P9 che duri per 1,5 secondi ed abbia una frequenza di 2 kHz:

```
FREQOUT 9, 1500, 2000
```

Programma di Esempio: TestPiezoWithFreqout.bs2

Questo programma esempio invia un tono di 2 kHz all’altoparlante collegato al pin I/O P9 per 1.5 secondi. Potete usare il Terminale di Debug per vedere quando l’altoparlante dovrebbe emettere un bip e quando dovrebbe smettere.

√ Digitate e lanciateTestPiezoWithFreqout.bs2.
√ Verificate che l’altoparlante emetta un tono chiaramente udibile durante il tempo in cui il Terminale di Debug visualizza il messaggio “Emissione del Tono…”

```
' Che cosa è un Microcontrollore- TestPiezoWithFreqout.bs2
' Emette un tono dall'altoparlante con il comando FREQOUT.

'{$STAMP BS2}
'{$PBASIC 2.5}

DEBUG "Emissione del Tono...", CR

FREQOUT 9, 1500, 2000

DEBUG "Tono Emesso."
```

Il Vostro Turno – Regolazione della Frequenza e della Durata.

√ Salvate TestPiezoWithFreqout.bs2 con un nome diverso.
√ Provate con valori diversi degli argomenti ***Duration*** e ***Freq1***.
√ Dopo ciascun cambiamento, lanciate il programma ed annotate l’effetto.
√ Quando l’argomento ***Freq1*** diventa più grande, il tono aumenta o diminuisce di frequenza? Per rispondere a questa domanda, provate valori di 1500, 2000, 2500 e 3000.

ESERCIZIO #2: SUONI DI ANIMAZIONE

Molti giocattoli contengono dei microcontrollori che vengono usati per creare dei suoni di animazione. I suoni di animazione implicano la variazione rapida di frequenza emessa dall’altoparlante. Potrete inoltre ottenere effetti interessanti miscelando due toni diversi con l’uso dell’argomento opzionale ***Freq2*** del comando **FREQOUT**. Questo Esercizio spiega tutte e due le tecniche.

Programmare i Suoni di Animazione

I suoni di animazione hanno tre diverse componenti:

1. Intervallo
2. Durata
3. Frequenza

L'intervallo è il tempo che intercorre tra l'emissione dei toni, e per creare questo intervallo potete usare il comando **PAUSE**. La durata è la quantità di tempo in cui un tono viene emesso, che voi potete impostare usando l'argomento ***Duration*** del comando **FREQOUT**. La frequenza determina l'acutezza del tono emesso. Maggiore la frequenza, più acuto è il tono emesso, minore la frequenza, più grave è il tono emesso. Questo, è ovviamente determinato dall'argomento ***Freq1*** del comando **FREQOUT**.

Programma di Esempio: ActionTones.bs2

ActionTones.bs2 mostra alcune diverse combinazioni di pausa, durata e frequenza. La prima sequenza di toni suona in modo simile ad una sveglia digitale. La seconda suona in modo molto simile ad un robot di un comune film di fantascienza. La terza è più simile agli effetti sonori dei vecchi video giochi.

√ Caricate e Lanciate ActionTones.bs2.

```
' Che cosa è un Microcontrollore- ActionTones.bs2
' Dimostrazione di come diverse combinazione di pausa, durata e frequenza
' possano essere usate per creare effetti sonori.

'{$STAMP BS2}
'{$PBASIC 2.5}

duration       VAR     Word
frequency      VAR     Word

DEBUG "Alarm...", CR
   PAUSE 100
   FREQOUT 9, 500, 1500
   PAUSE 500
   FREQOUT 9, 500, 1500
   PAUSE 500
   FREQOUT 9, 500, 1500
   PAUSE 500
   FREQOUT 9, 500, 1500
   PAUSE 500
```

```
DEBUG "Robot replicante...", CR
   PAUSE 100
   FREQOUT 9, 100, 2800
   FREQOUT 9, 200, 2400
   FREQOUT 9, 140, 4200
   FREQOUT 9,  30, 2000
   PAUSE 500

DEBUG "Iperspazio...", CR
  PAUSE 100
  FOR duration = 15 TO 1 STEP 1
    FOR frequency = 2000 TO 2500 STEP 20
      FREQOUT 9, duration, frequency
    NEXT
  NEXT

DEBUG "Fatto", CR

END
```

Come funziona ActionTones.bs2

La routine "Alarm" suona in maniera simile ad una sveglia digitale. Questa routine emette dei toni alla frequenza fissa di 1.5 kHz per la durata di 0.5 s con un ritardo fisso tra i toni di 0.5 s. La routine "Robot replicante" usa varie frequenze con durate brevi.

La routine "Iperspazio" non usa ritardo, ma cambia sia la durata che la frequenza. Usando dei cicli **FOR…NEXT** per variare rapidamente la frequenza e la durata, potrete ottenere interessanti effetti sonori. Quando un ciclo **FOR…NEXT** viene eseguito all'interno di un altro ciclo, viene chiamato ciclo nidificato. Viene qui indicato come funziona il ciclo nidificato **FOR…NEXT** mostrato sotto. La variabile **duration** inizia da 15, quindi il ciclo **frequency** inizia ad inviare all'altoparlante piezo frequenze di 2000, quindi 2020, poi 2040, e così via fino a 2500. quando il ciclo frequency è terminato, il ciclo **duration** ha appena eseguito uno dei suoi 15 passaggi. Così sottrae uno dal valore di **duration** e ripete di nuovo il ciclo **frequency**.

```
FOR duration = 15 TO 1
  FOR frequency = 2000 TO 2500 STEP 15
    FREQOUT 9, duration, frequency
  NEXT
NEXT
```

Programma di Esempio: NestedLoops.bs2

Per capire meglio come funzionano i cicli **FOR…NEXT** annidati, NestedLoops.bs2 usa il comando **DEBUG** per mostrare il valore di una versione molto meno complicata di un ciclo annidato usato in ActionTones.bs2.

√ Caricate e Lanciate NestedLoops.bs2.
√ Esaminate l'uscita del Terminale di Debug e verificate come cambiano gli argomenti durata e frequenza ad ogni iterazione del ciclo.

```
' Che cosa è un Microcontrollore- NestedLoops.bs2
' Dimostrazione del funzionamento del ciclo nidificato in ActionTones.bs2.

'{$STAMP BS2}
'{$PBASIC 2.5}

duration       VAR     Word
frequency      VAR     Word

DEBUG "Durata     Frequenza", CR,
      "--------   ---------", CR

FOR duration = 4000 TO 1000 STEP 1000
  FOR frequency = 1000 TO 3000 STEP 500
    DEBUG "   "  , DEC5 duration,
          "       ", DEC5 frequency, CR
    FREQOUT 9, duration, frequency
  NEXT
  DEBUG CR
NEXT

END
```

Il Vostro Turno – Altri Effetti Sonori

Chiaramente ci sono un numero quasi infinito di modi per modificare ActionTones.bs2 per ottenere diverse combinazioni di suoni. Viene qui mostrata solo una delle modifiche alla routine "Iperspazio":

```
DEBUG "Salto nell'Iperspazio...", CR
  FOR duration = 15 TO 1 STEP 3
    FOR frequency = 2000 TO 2500 STEP 15
      FREQOUT 9, duration, frequency
    NEXT
  NEXT
  FOR duration = 1 TO 36 STEP 3
```

```
    FOR frequency = 2500 TO 2000 STEP 15
      FREQOUT 9, duration, frequency
    NEXT
  NEXT
```

√ Salvate il vostro programma con il nuovo nome ActionTonesYourTurn.bs2.
√ Divertitevi con questa ed altre modifiche di vostra invenzione.

Due frequenze insieme

Potete inviare due frequenze nello stesso momento. In audio, questo viene chiamato "miscelazione". Ricordate la sintassi del comando **FREQOUT** dall'Esercizio #1:

FREQOUT ***Pin, Duration, Freq1 {, Freq2}***

Potete usare l'argomento opzionale ***Freq2*** per miscelare due frequenze con il comando **FREQOUT**. Per esempio, potete miscelare insieme 2 khz e 3 khz in questo modo:

```
FREQOUT 9, 1000, 2000, 3000
```

Le tastiere dei telefoni multifrequenza sono un altro esempio di due frequenze miscelate insieme. Nelle telecomunicazioni, questo viene chiamato DTMF (Dual Tone Multi Frequency). Esiste anche un comando PBASIC chiamato **DTMFOUT** che è stato pensato proprio per inviare toni telefonici. Per esempi di progetti dove viene usata la composizione di numeri telefonici, vedere il comando **DTMFOUT** nel Manuale del BASIC Stamp.

Programma di Esempio: MixingTones.bs2

Questo programma esempio spiega la differenza nei toni che ottenete miscelando insieme 2 khz e 3 khz. Viene anche spiegato un fenomeno interessante che accade quando vengono miscelate due forme d'onda che sono molto vicine tra loro in frequenza. Quando miscelate 2000 hz con 2001 hz, il tono si affievolirà e riaumenterà di intensità una volta al secondo (alla frequenza di 1 hz). Se miscelate 2000 Hz con 2002 Hz, lo farà due volte al secondo (2 Hz), e così via.

Si definisce Battimento quando due toni di frequenza simile vengono suonati insieme causando un'affievolimento ed un rafforzamento dell'intensità. La frequenza di quell'affievolimento e del rafforzamento è la differenza delle due frequenze. Se la differenza è 1 Hz, il tono "batterà" ad 1 Hz. Se la differenza è 2 Hz, il tono batterà a 2 Hz.

Le variazioni della pressione dell'aria create dall'altoparlante piezoelettrico sono chiamate onde sonore. Quando il tono è al suo massimo, le variazioni della pressione dell'aria causate dalle due frequenze si sommano (si chiama sovrapposizione). Quando il tono è al suo minimo, le variazioni della pressione dell'aria si cancellano vicendevolmente (si chiama interferenza).

√ Caricate e Lanciate MixingTones.bs2.
√ Quando i toni vengono emessi, guardate il Terminale di Debug, annotate la diversità di effetti che si verificano miscelando toni diversi.

```
' Che cosa è un Microcontrollore- MixingTones.bs2
' Dimostrazione delle cose che accadono miscelando due toni.

'{$STAMP BS2}
'{$PBASIC 2.5}

DEBUG "Frequency = 2000", CR
FREQOUT 9, 4000, 2000

DEBUG "Frequency = 3000", CR
FREQOUT 9, 4000, 3000

DEBUG "Frequency = 2000 + 3000", CR
FREQOUT 9, 4000, 2000, 3000

DEBUG "Frequency = 2000 + 2001", CR
FREQOUT 9, 4000, 2000, 2001

DEBUG "Frequency = 2000 + 2002", CR
FREQOUT 9, 4000, 2000, 2002

DEBUG "Frequency = 2000 + 2003", CR
FREQOUT 9, 4000, 2000, 2003

DEBUG "Frequency = 2000 + 2005", CR
FREQOUT 9, 4000, 2000, 2005

DEBUG "Frequency = 2000 + 2010", CR
FREQOUT 9, 4000, 2000, 2010

DEBUG "Done", CR

END
```

Il Vostro Turno – Minimizzare il Codice

MixingTones.bs2 è stato scritto per dimostrare alcuni aspetti interessanti che accadono quando miscelate due diverse frequenze usando l'argomento opzionale ***Freq2*** del comando **FREQOUT**. Sebbene sia estremamente inefficiente.

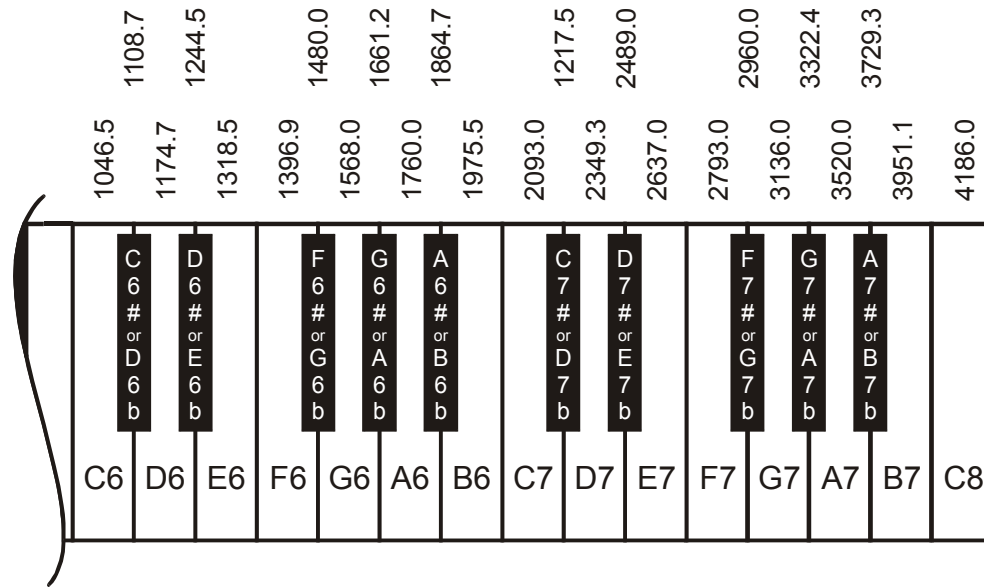
√ Modificate MixingTones.bs2 in modo che l'argomento ***Freq2*** vada ciclicamente da 2001 a 2005 usando una variabile word ed un ciclo.

ESERCIZIO #3: NOTE MUSICALI E CANZONI SEMPLICI

La Figura 8-3 mostra i 25 tasti più a destra della tastiera di un pianoforte. Mostra anche le frequenze a cui vibra quella corda del piano quando il tasto corrispondente viene battuto. I tasti e le note corrispondenti sono etichettati da C6 fino a C8 (viene usata la notazione musicale anglosassone). Questi tasti sono raggruppati in gruppi di 12 chiamati ottave. Quando contate 12 tasti, assicuratevi di comprendere i tasti neri. Inoltre la sequenza di note viene ripetuta ogni 12 tasti. Da notare che C7 ha una frequenza doppia rispetto a C6. E questo è vero per ciascuna nota dell'ottava successiva. Similmente, se scendete di un'ottava, la frequenza sarà la metà.

Ricerca Internet di "scala musicale": usando le parole "scala musicale" in un motore di ricerca come Google o Yahoo, troverete moltissime informazioni affascinanti circa la storia, la fisica e la psicologia del soggetto. La scala musicale con 12 note è la scala principale della musica occidentale. Altre culture usano scale che possono contenere da 2 a 35 note per ottava.

Se avete mai ascoltato un cantante allenarsi con il solfeggio, "Do Re Mi Fa Sol La Si Do", noterete che il cantante sta provando ad eseguire le note che si ottengono battendo sui tasti bianchi di un pianoforte. Questi tasti bianchi sono chiamati tasti naturali. Un tasto nero in un piano si può chiamare sia diesis che bemolle. Per esempio, il tasto nero tra i tasti C e D si può chiamare sia C-diesis ($C^{\#}$) che D-bemolle (D^{b}). Che un tasto sia chiamato diesis o bemolle dipende dal pezzo particolare che viene suonato, ed è meglio lasciare che queste regole siano spiegate in un corso di musica.

Figura 8-3
Parte destra della Tastiera del Pianoforte e Frequenze Associate

Sistema di Accordo: La tastiera della Figura 8-3 usa un metodo di accordo chiamato scala temperata. Le frequenze vengono determinate usando una nota di riferimento, quindi la si moltiplica per $2^{(n/12)}$ per valori di n = 1, 2, 3, etc. Per esempio prendete la frequenza di A6 e moltiplicatela per $2^{(1/12)}$ per ottenere la frequenza di A6#. Quindi moltiplicate quest'ultima per $2^{(2/12)}$ ed otterrete la frequenza di B6, e così via. Come viene mostrato nell'esempio seguente a partire dalla frequenza di A6 come riferimento:

La frequenza di A6 è 1760

$2^{(2/12)}$ = 1.1224

1760 X 1.224 = 1975.5

1975.5 è la frequenza di B6

Programmazione delle Note Musicali

Il comando **FREQOUT** è utile anche per le note musicali. Perché il BASIC Stamp sia programmato per suonare musica con un altoparlante piezo, implica seguire una varietà

di regole usate per suonare musica con qualsiasi altro strumento musicale. Queste regole si applicano agli stessi elementi che sono stati usati per creare effetti sonori, la frequenza, la durata e la pausa. Il prossimo esempio suona alcune delle frequenze delle note musicali sull'altoparlante piezo, ciascuna con una durata di mezzo secondo.

Programma di Esempio: DoReMiFaSolLaTiDo.bs2

√ Caricate e Lanciate DoReMiFaSolLaTiDo.bs2

```
' Che cosa è un Microcontrollore- DoReMiFaSolLaTiDo.bs2
' Invia le note di un'ottava per mezzo secondo ciascuna ad un altoparlante
' piezoelettrico.

'{$STAMP BS2}
'{$PBASIC 2.5}

'Solfege            Tone                    Note

DEBUG "Do...", CR:  FREQOUT 9,500,1047     ' C6

DEBUG "Re...", CR:  FREQOUT 9,500,1175     ' D6

DEBUG "Mi...", CR:  FREQOUT 9,500,1319     ' E6

DEBUG "Fa...", CR:  FREQOUT 9,500,1396     ' F6

DEBUG "Sol..", CR:  FREQOUT 9,500,1568     ' G6

DEBUG "La...", CR:  FREQOUT 9,500,1760     ' A6

DEBUG "Ti...", CR:  FREQOUT 9,500,1976     ' B6

DEBUG "Do...", CR:  FREQOUT 9,500,2093     ' C7

END
```

Il Vostro Turno – Note diesis/bemolle

√ Usate le frequenze mostrate in Figura 8-3 per aggiungere le cinque note diesis/bemolle a DoReMiFaSolLaTiDo.bs2

√ Modificate il vostro programma in modo che suoni l'ottava superiore successiva. Suggerimento: Vi risparmierete molta digitazione se userete l'operazione **`* 2`** dopo ciascun argomento ***`Freq1`***. Per esempio, **`FREQOUT 9, 500, 1175 * 2`** vi darà la nota D7, la nota D della 7ª ottava.

Memorizzare e Recuperare Sequenze di Note Musicali

Un buon modo di salvare le note musicali è di memorizzarle usando il comando EEPROM del BASIC Stamp. Sebbene per far questo potreste usare molti comandi, un modo migliore è usare la direttiva **`DATA`**. Questa è la sintassi per la direttiva **`DATA`**:

{Symbol} DATA *{Word} DataItem {, {Word} DataItem, ... }*

Qui c'è un esempio di come usare la direttiva **`DATA`** per memorizzare i caratteri che corrispondono alle note musicali.

```
Notes DATA "C","C","G","G","A","A","G"
```

Per recuperare questi caratteri potete usare il comando **`READ`**. La lettera 'C' è posizionata all'indirizzo **`Notes + 0`**, e la seconda lettera 'C' è posizionata in **`Notes + 1`**. Quindi, c'è una lettera 'G' in **`Notes + 2`**, e così via. Per esempio, se volete caricare l'ultima lettera 'G' in una variabile byte chiamata **`noteLetter`**, usate il comando:

```
READ Notes + 6, noteLetter
```

I caratteri sono memorizzati come byte. Ciascun carattere ha un codice numerico con cui viene memorizzato. Questi codici vengono chiamati codici ASCII, e sono stati presentati nel Capitolo #1. Questi valori possono andare da 0 a 255.

Ci vuole più di un byte per memorizzare un valore maggiore di 255. La variabile word occupa in memoria lo spazio di due byte, ed una variabile word può memorizzare un numero da 0 e 65535.

Con la direttiva **`DATA`** potete anche memorizzare elenchi di numeri. Il valore della frequenza e della durata che il BASIC Stamp usa per le note musicali devono essere memorizzate in variabili word perché di solito sono più grandi di 255. Di seguito viene spiegato come farlo con la direttiva **`DATA`**.

```
Frequencies DATA Word 2093, Word 2093, Word 3136, Word 3136,
                 Word 3520, Word 3520, Word 3136
```

A causa del fatto che ciascuno di questi valori occupano lo spazio di due byte, il recupero di questi con il comando read è diverso dal recupero dei caratteri. Il primo 2093 è in **`Frequencies + 0`**, ma il secondo 2093 è posizionato in **`Frequencies + 2`**. Il primo 3136 è posizionato in **`Frequencies + 4`**, ed il secondo 3136 è posizionato in **`Frequencies + 6`**.

I valori nella direttiva Frequencies DATA corrispondono con le note musicali della direttiva **Notes DATA**.

Viene qui indicato un ciclo **FOR…NEXT** che mette le **Notes DATA** in una variabile chiamata **noteLetter**, quindi mette le **Frequencies DATA** in una variabile di nome **noteFreq**.

```
FOR index = 0 to 6
  READ Notes + index, noteLetter
  READ Frequencies + (index * 2), Word noteFreq
  DEBUG noteLetter, "  ", DEC noteFreq, CR
NEXT
```

Che cosa fa (index * 2)? Ciascun valore memorizzato nella direttiva **Frequencies DATA** occupa una word (due byte), mentre ciascun carattere nella direttiva**Notes DATA** occupa solamente un byte. Il valore di **index** aumenta di uno ad ogni iterazione del ciclo **FOR…NEXT**. È perfetto per recuperare i caratteri delle note usando il comando **READ Notes + index, noteLetter**. Il problema è che per ogni byte in **Notes**, la variabile **index** deve puntare ad una posizione due indirizzi più in avanti nell'elenco delle **Frequencies**. Il comando **READ Frequencies + (index * 2), Word noteFreq**, si occupa di questo.

Il prossimo programma esempio memorizza le note e la loro durata usando il comando **DATA**, e per suonare ciascuna frequenza di nota per una specifica durata usa il comando **FREQOUT**. Il risultato sono le prime cinque note della canzone per bambini "Twinkle Twinkle Little Star".

La canzone dell'alfabeto usata dai bambini per memorizzare l'alfabeto "ABC" usa le stesse note di "Twinkle Twinkle Little Star".

Programma di Esempio: TwinkleTwinkle.bs2

Questo programma esempio mostra come usare la direttiva **DATA** per memorizzare elenchi ed usa il comando **READ** per richiamare i valori degli elenchi.

√ Caricate e Lanciate TwinkleTwinkle.bs2
√ Verificate che le note siano simili alla canzone "Twinkle Twinkle Little Star".

√ Usate il Terminale di Debug per verificare che funzioni come desiderato recuperando e visualizzando i valori dalle tre direttive **DATA**.

```
' Che cosa è un Microcontrollore- TwinkleTwinkle.bs2
' Suona le prime sette note della canzone “Twinkle Twinkle Little Star”.

'{$STAMP BS2}
'{$PBASIC 2.5}

Notes            DATA  "C","C","G","G","A","A","G"

Frequencies      DATA  Word 2093, Word 2093, Word 3136, Word 3136,
                       Word 3520, Word 3520, Word 3136

Durations        DATA  Word 500, Word 500, Word 500, Word 500,
                       Word 500, Word 500, Word 1000

index            VAR   Nib
noteLetter       VAR   Byte
noteFreq         VAR   Word
noteDuration     VAR   Word

DEBUG    "Nota  Durata  Frequenza", CR,
         "----  --------  ---------", CR

FOR index = 0 TO 6

  READ Notes + index, noteLetter
  DEBUG "   ", noteLetter

  READ Durations + (index * 2), Word noteDuration
  DEBUG "      ", DEC4 noteDuration

  READ Frequencies + (index * 2), Word noteFreq
  DEBUG "       ", DEC4 noteFreq, CR

  FREQOUT 9, noteDuration, noteFreq

NEXT

END
```

Il Vostro Turno – Aggiungere e Suonare più Note

Questo programma suona le prime sette note della canzone “Twinkle Twinkle Little Star”. Le parole sono “Twin-kle twin-kle lit-tle star”. La frase successiva della canzone è “How I won-der what you are”, e le sue note sono F, F, E, E, D, D, C. come per la prima

frase, l'ultima nota dura il doppio delle altre note. Per aggiungere questa frase alla canzone dal programma TwinkleTwinkle.bs2, dovrete espandere appropriatamente ciascuna direttiva **DATA**. Non dimenticate di cambiare il ciclo **FOR…NEXT** in modo che vada da 0 a 13 invece che da 0 a 6.

√ Modificate TwinkleTwinkle.bs2 in modo che suoni le prime due strofe della canzone invece che solamente la prima.

ESERCIZIO #4: MUSICA CON IL MICROCONTROLLORE

La durata delle note non è descritta sullo spartito in termini di millisecondi. Invece, sono descritte come interi, mezzi, quarti, ottavi, sedicesimi, trentaduesimi. Come suggerito dal nome, una mezza nota dura la metà di una nota intera. Una quarta dura un quarto di una nota intera e così via. Quanto è lunga una nota intera? Dipende dal pezzo musicale che si sta suonando. Un pezzo può essere suonato in un tempo in cui una nota intera dura quattro secondi, un altro pezzo può avere una nota intera di due secondi, ed un altro ancora può avere qualsiasi altra durata.

Le pause sono il tempo tra le note in cui nessun tono viene suonato. Anche le pause sono misurate in termini di interi, mezzi, quarti, ottavi, sedicesimi e trentaduesimi di secondo.

Un Sistema Migliore per Memorizzare e Recuperare la Musica

Potete scrivere programmi che memorizzeranno il doppio della musica sul vostro BASIC Stamp usando i byte invece delle word nelle vostre direttive **DATA**. Potrete anche modificare il vostro programma per rendere le note musicali più facilmente leggibili usando alcune delle più comuni convenzioni musicali per le note e la durata. Questo Esercizio comincerà con la spiegazione di come memorizzare le informazioni musicali in un modo più strettamente correlate ai concetti di note, durata e pause. Verrà anche spiegato il concetto di Tempo, che sarà rivisto anche nell'esercizio successivo.

Viene qui mostrato un esempio di direttiva **DATA** che memorizza le note musicali e la durata del prossimo programma esempio. Quando suonato, dovrebbe assomigliare alla canzone "Frere Jacques". Nella direttiva **Notes DATA** verranno memorizzate solamente I caratteri che rappresentano le note perché i comandi **LOOKUP** e **LOOKDOWN** si incaricheranno di associare i caratteri alle loro frequenze corrispondenti.

```
Notes           DATA    "C","D","E","C","C","D","E","C","E","F",
                        "G","E","F","G","Q"
```

```
Durations        DATA      4,  4,  4,  4,  4,  4,  4,  4,  4,  4,
                           2,  4,  4,  2

WholeNote        CON     2000
```

Il primo numero nella direttiva **`Durations DATA`** dice al programma quanto a lungo deve suonare la prima nota nella direttiva **`Notes Data`**. La seconda durata è per la seconda nota, e così via. Le durate non sono più espresse in termini di millisecondi. Invece, sono numeri molto più piccoli che possono essere memorizzati in byte, così nella direttiva **`Data`** non c'è più il prefisso **`Word`**. Rapportati alla memorizzazione dei valori in termini di millisecondi, questi numeri sono più strettamente correlati alla notazione musicale. Viene qui elencata una lista del significato di ciascuna durata.

- 1 – Quatttro Quarti
- 2 – Due Quarti
- 4 – Un Quarto
- 8 – Un Ottavo
- 16 – Un Sedicesimo
- 32 – Un Trentaduesimo

8

Dopo aver letto ciascun valore **`Durations`** dalla direttiva **`DATA`**, viene usato per dividere il valore **`WholeNote`** per ottenere il valore ***`Duration`*** usato nel commando **`FREQOUT`**. La quantità di tempo di durata di ciascuna nota dipende dal tempo della canzone. Un tempo più veloce significa che ciascuna nota dura per meno tempo, mentre un tempo più lento significa che una nota dura più a lungo. Dal momento che la durata di tutte le note è una frazione di nota intera, per impostare il tempo potrete usare la durata della nota intera.

Che cosa significa la "`Q`" in `Notes DATA`? "Q" sta per quit (in inglese lascia), ed un ciclo **`DO WHILE…LOOP`** controlla ad ogni iterazione la presenza del carattere "Q".

Come faccio a suonare una pausa? Potete inserire una pausa tra le note inserendo il carattere **`"P"`**. Nella sezione Il Vostro Turno ci sono le prime cinque note dalla 5th sinfonia di Beethoven, in cui è presente una pausa.

Come faccio a suonare delle note diesis/bemolle? Il programma NotesAndDurations.bs2 nelle sue tabelle di lookup ha anche i valori per le note diesis/bemolle. Quando usate per la nota la lettera minuscola, verrà suonata la nota normale. Per esempio, se volete suonare la nota B normale, usate "b" invece di "B". Ricordate che essa ha la stessa frequenza della A diesis.

Programma di Esempio: NotesAndDurations.bs2

- √ Caricate e Lanciate NotesAndDurations.bs2.
- √ In che modo suona?

```
' Che cosa è un Microcontrollore- NotesAndDurations.bs2
' Suona le prime note di Frere Jacques.

'{$STAMP BS2}
'{$PBASIC 2.5}

Notes           DATA    "C","D","E","C","C","D","E","C","E","F",
                        "G","E","F","G","Q"

Durations       DATA      4,  4,  4,  4,  4,  4,  4,  4,  4,  4,
                          2,  4,  4,  2

WholeNote       CON     2000

index           VAR     Byte
offset          VAR     Nib

noteLetter      VAR     Byte
noteFreq        VAR     Word
noteDuration    VAR     Word

DO UNTIL noteLetter = "Q"

  READ Notes + index, noteLetter

  LOOKDOWN noteLetter, [  "A",  "b",  "B",  "C",  "d",
                          "D",  "e",  "E",  "F",  "g",
                          "G",  "a",  "P",  "Q"     ], offset

  LOOKUP offset,       [ 1760, 1865, 1976, 2093, 2217,
                         2349, 2489, 2637, 2794, 2960,
                         3136, 3322,    0,    0     ], noteFreq

  READ Durations + index, noteDuration

  noteDuration = WholeNote / noteDuration

  FREQOUT 9, noteDuration, noteFreq

  index = index + 1

LOOP

END
```

Come funziona NotesAndDurations.bs2

Le direttive **DATA, Notes** e **Durations** sono già state spiegate prima del programma. Queste direttive insieme con la costante **WholeNote** sono usate per memorizzare tutti i dati musicali usati dal programma.

Le dichiarazioni delle cinque variabili usate nel programma sono mostrate sotto. Sebbene non venga più usato un ciclo **FOR…NEXT** per accedere ai dati, ci deve ancora essere una variabile (**index**) che tenga traccia di quale **DATA** viene letta nelle **Notes** e **Durations**. La variabile **offset** è usata nei comandi **LOOKDOWN** e **LOOKUP** per selezionare un particolare valore. La variabile **noteLetter** memorizza un carattere variable ricavato dal commando **READ**. Poi sono usati i comandi **LOOKUP** e **LOOKDOWN** per convertire questo carattere in un valore di frequenza. Questo valore viene memorizzato nella variabile **noteFreq** ed usato come argomento ***Freq1*** del commando **FREQOUT**. La variabile **noteDuration** è usata in un commando **READ** per ricevere un valore dal **DATA Durations**. Viene anche usato per calcolare ***Duration*** usata nel commando **FREQOUT**.

8

```
index         VAR  Byte
offset        VAR  Nib

noteLetter    VAR  Byte
noteFreq      VAR  Word
noteDuration  VAR  Word
```

Il ciclo principale continua ad essere eseguito fino alla lettura di ‘Q’ nel **DATA Notes**.

```
DO UNTIL noteLetter = "Q"
```

Un commando **READ** legge un carattere dal **DATA Notes**, e lo memorizza nella variabile **noteLetter**. La variabile **noteLetter** viene quindi usata nel commando **LOOKDOWN** per impostare il valore della variabile **offset**. Ricordate che **offset** memorizza un 1 se viene rilevato “b”, un 2 se viene rilevato “B”, un 3 se viene rilevato “C”, e così via. Questo valore **offset** viene quindi usato in un comando **LOOKUP** per decidere quale dovrebbe essere il valore della variabile **noteFreq**. Se **offset** è 1, **noteFreq** sarà 1865, se **offset** è 2, **noteFreq** sarà 1976, se offset è 3, **noteFreq** è 2093, e così via.

```
READ Notes + index, noteLetter

LOOKDOWN noteLetter, [  "A",  "b",  "B",  "C",  "d",
                        "D",  "e",  "E",  "F",  "g",
                        "G",  "a",  "P",  "Q" ], offset

LOOKUP offset,       [ 1760, 1865, 1976, 2093, 2217,
```

```
                                        2349, 2489, 2637, 2794, 2960,
                                        3136, 3322,    0,    0  ], noteFreq
```

La frequenza della nota è stata determinata, ma deve ancora essere scelta la durata. Il comando **READ** usa il valore di **index** per mettere un valore dalla **DATA Durations** nella variabile **noteDuration**.

```
      READ Durations + index, noteDuration
```

Quindi, **noteDuration** è impostata uguale alla costante **WholeNote, e** divisa per **noteDuration**. Se da un comando **READ** viene fuori essere 4, diventa 2000 ÷ 4 = 500. Se **noteDuration** è 8, diventa 1500 ÷ 8 = 250.

```
      noteDuration = WholeNote / noteDuration
```

Ora che **noteDuration** e **noteFreq** sono state determinate, il comando **FREQOUT** suona la nota.

```
      FREQOUT 9, noteDuration, noteFreq
```

Ad ogni iterazione del ciclo principale, deve essere aumentata di uno la variabile **index**. Quando il ciclo principale torna all'inizio, la prima cosa che il programma fa è leggere la nota seguente, usando la variabile **index**.

```
      index = index + 1

    LOOP
```

Il Vostro Turno – Sperimentare con il Tempo e Differenti Accordi.

La durata di ciascuna nota è legata al Tempo. Potete cambiare il Tempo regolando la costante **WholeNote**. Se la aumentate a 2250, il Tempo diminuirà, e la canzone suonerà più lentamente. Se la diminuite a 1750, il Tempo aumenterà e la canzone suonerà più velocemente.

- √ Salvate il programma NotesAndDurations.bs2 con il nuovo nome di NotesAndDurationsYourTurn.bs2.
- √ Modificate il Tempo di NotesAndDurationsYourTurn.bs2 regolando il valore di **WholeNote**. Provate valori di 1500, 1750, 2000, e 2250.
- √ Rilanciate il programma dopo ciascuna modifica, e decidete quale variazione suona meglio.

Inserire dati musicali è più facile quando tutto quello che dovete fare è registrare note e durate. Di seguito sono elencate le prime otto note della quinta sinfonia di Beethoven.

```
Notes     DATA "G","G","G","e","P","F","F","F","D","Q"

Durations DATA   8,  8,  8,  2,  8,  8,  8,  8,  2

WholeNote CON   2000
```

√ Salvate il vostro programma modificato Beethoven'sFifth.bs2.
√ Sostituite le direttive **`DATA Notes`** e **`Durations`** e la dichiarazione di costante **`WholeNote`** con il codice scritto sopra.
√ Lanciate il programma. Non vi suona familiare?

Aggiungere Caratteristiche Musicali

Il programma esempio che avete appena finito ha spiegato le note, la durata e le pause. Ha anche usato la durata di una nota intera per determinare il Tempo. I telefoni cellulari che suonano musichette per avvisarvi che qualcuno vi sta chiamando hanno tre caratteristiche non supportate dal programma esempio precedente:

- Essi suonano note "punteggiate".
- Determinano la durata della nota intera da un valore chiamato Tempo.
- Suonano note più estese che 1 ottava.

Il termine "punteggiate" si riferisce ad un punto usato negli spartiti per indicare che una nota deve essere suonata una volta e mezzo la sua normale durata. Per esempio, un quarto di nota punteggiata, deve essere suonata per una durata di un quarto di nota più un ottavo di nota. Una mezza nota punteggiata dura per una mezza nota più un quarto di nota. Potete aggiungere una tabella che memorizzi se una nota è punteggiata o No. In questo esempio, uno zero significa che non c'è punteggiatura, mentre un 1 significa che c'è punteggiatura:

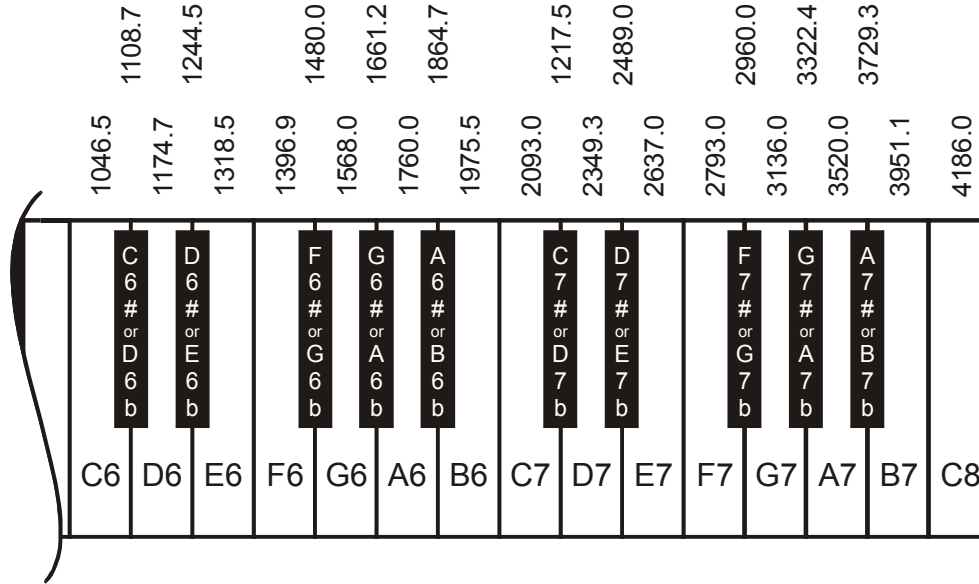
```
Dots            DATA      0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,
                          0,  0,  0,  1,  0
```

I telefoni cellulari tipicamente prendono il Tempo di una canzone in battute al minuto. Questo è lo stesso che dire "quarti di nota al minuto"(i famosi quattro quarti).

```
BeatsPerMin   CON  200
```

La Figura 8-4 è una ripetizione della Figura 8-3 dalla pagina 233. Mostra la 6ª e la 7ª ottava della tastiera del piano. Queste sono le ottave che suonano meglio quando suonate con un altoparlante piezo. Di seguito c'è un esempio di direttiva **`DATA`** che userete nella sezione Il Vostro Turno per suonare note per più di un'ottava.

```
Octaves         DATA      6,  7,  6,  6,  6,  6,  6,  6,  6,  7,  6,
                          6,  6,  6
```

Figura 8-4
Rightmost Piano Keys and Their Frequencies

Programma di Esempio: MusicWithMoreFeatures.bs2

Questo programma esempio suona le rime note di “For He’s a Jolly Good Yellow”. Tutte le note appartengono alla stessa ottava la (7ª), ma alcune delle note sono punteggiate. Nella sezione Il Vostro Turno, proverete un esempio che usa le note per più di un’ottava, e le note punteggiate.

√ Caricate e Lanciate MusicWithMoreFeatures.bs2.
√ Contate le note e vedete se potete sentire le note punteggiate (1 ½ la durata).
√ Ascoltate anche per sentire se le note appartengono alla settima ottava. Provate a cambiare una di queste note con la corrispondente della sesta ottava. La variazione nel modo in cui la musica suona, è abbastanza drastico.

```
' Che cosa è un Microcontrollore- MusicWithMoreFeatures.bs2
```

```
' Suona l'inizio di "For He's a Jolly Good Fellow".

'{$STAMP BS2}
'{$PBASIC 2.5}

Notes           DATA    "C","E","E","E","D","E","F","E","E","D","D",
                        "D","C","D","E","C","Q"
Octaves         DATA      7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,
                          7,  7,  7,  7,  7
Durations       DATA      4,  2,  4,  4,  4,  4,  2,  2,  4,  2,  4,
                          4,  4,  4,  2,  2
Dots            DATA      0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,
                          0,  0,  0,  1,  0

BeatsPerMin     CON     320

index           VAR     Byte
offset          VAR     Nib

noteLetter      VAR     Byte
noteFreq        VAR     Word
noteDuration    VAR     Word
noteOctave      VAR     Nib
noteDot         VAR     Bit

wholeNote       VAR     Word
wholeNote = 60000 / BeatsPerMin * 4

DO UNTIL noteLetter = "Q"

  READ Notes + index, noteLetter

  LOOKDOWN noteLetter,  [ "C",  "d",  "D",  "e",  "E",
                          "F",  "g",  "G",  "a",  "A",
                          "b",  "B",  "P",  "Q"     ], offset

  LOOKUP offset,       [ 4186, 4435, 4699, 4978, 5274,
                         5588, 5920, 6272, 6645, 7040,
                         7459, 7902,    0,    0     ], noteFreq

  READ Octaves + index, noteOctave
  noteOctave = 8 - noteOctave
  noteFreq = noteFreq / (DCD noteOctave)

  READ Durations + index, noteDuration
  noteDuration = WholeNote / noteDuration

  READ Dots + index, noteDot
  IF noteDot = 1 THEN noteDuration = noteDuration * 3 / 2

  FREQOUT 9, noteDuration, noteFreq
```

```
  index = index + 1

LOOP

END
```

Come funziona MusicWithMoreFeatures.bs2

Sotto elencata ci sono i dati musicali dell'intera canzone. Per ciascuna nota nella direttiva **DATA Notes**, c'è il valore corrispondente delle direttive **DATA Octaves, Durations,** e **Dots**. Per esempio, la prima nota è una C nella 7[a] ottava; è un quarto di nota e non è punteggiata. Qui c'è un altro esempio: la pen'ultima nota (non includendo la "Q") è una nota E, nella 7[a] ottava. È una mezza nota, ed è punteggiata. C'è anche una costante **BeatsPerMin** che imposta il Tempo della canzone.

```
Notes           DATA    "C","E","E","E","D","E","F","E","E","D","D",
                        "D","C","D","E","C","Q"
Octaves         DATA      7,  7,  7,  7,  7,  7,  7,  7,  7,  7,  7,
                          7,  7,  7,  7,  7
Durations       DATA      4,  2,  4,  4,  4,  4,  2,  2,  4,  2,  4,
                          4,  4,  4,  2,  2
Dots            DATA      0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,
                          0,  0,  0,  1,  0

BeatsPerMin     CON     320
```

Nel programma esempio precedente, **WholeNote** era una costante. In questo caso, è una variabile che conterrà la durata di una nota intera in ms. dopo aver calcolato questo valore, **WholeNote** sarà usato per determinare la durata di tutte le alter note, proprio come nel programma precedente. Le variabili **index**, **offset**, **noteLetter**, e **noteDuration** sono usate nello stesso modo del programma precedente. la variabile **noteFreq** viene trattata in modo leggermente diverso dal momento che deve essere adattata all'ottava in cui la nota deve essere suonata. Le variabili **noteOctave** e **noteDot** sono state aggiunte per gestire le caratteristiche di ottava e punteggiatura.

```
wholeNote     VAR  Word

index         VAR  Byte
offset        VAR  Nib

noteLetter    VAR  Byte
noteFreq      VAR  Word
noteDuration  VAR  Word
```

```
noteOctave    VAR  Nib
noteDot       VAR  Bit
```

La variabile **wholeNote** viene calcolata usando **BeatsPerMin**. Il Tempo della canzone è definito in battute al minuto, ed il programma deve dividere 60000 per **BeatsPerMin,** quindi moltiplicarlo per 4. il risultato è il valore giusto di una nota intera.

```
wholeNote = 60000 / BeatsPerMin * 4
```

Le operazioni matematiche sono eseguite da sinistra a destra. Nel calcolo di **wholeNote = 60000 / beatsPerMin * 4**, il BASIC Stamp prima calcola **60000 / beatsPerMin**. Quindi, moltiplica quel risultato per 4.

Per raggruppare le operazioni possono essere usate le parentesi. Se volete fare prima la moltiplicazione beatsPerMin per 4, potete farlo così: **wholeNote = 60000 / (beatsPerMin * 4)**.

Questo è lo stesso del programma precedente:

```
DO UNTIL noteLetter = "Q"

  READ Notes + index, noteLetter

  LOOKDOWN noteLetter,  [ "C",  "d",  "D",  "e",  "E",
                          "F",  "g",  "G",  "a",  "A",
                          "b",  "B",  "P",  "Q"     ], offset
```

Ora che le note sono mischiate, la parte di codice che sceglie la frequenza della nota è cambiata. La tabella del comando **LOOKUP** dei valori, contiene le note dell'8ª ottava. Questi valori possono essere divisi, per 1 se volete suonare le note nell'8ª ottava, per 2 se volete suonare le note della 7ª ottava, per 4 se volete suonare le note della 6ª ottava e per 8 se volete suonare le note della 5ª ottava. A questo segue la divisione. Tutto quello che il comando **LOOKUP** fa, è mettere nella variabile **noteFreq** una nota della 8ª ottava.

```
  LOOKUP offset,       [ 4186, 4435, 4699, 4978, 5274,
                         5588, 5920, 6272, 6645, 7040,
                         7459, 7902,    0,    0     ], noteFreq
```

Qui viene spiegato come la variabile **noteFreq** è aggiustata per l'ottava corretta. Primo, il comando **READ** prende il valore di ottava memorizzato in **DATA Octaves**. Questo può essere un valore da 5 ad 8.

```
  READ Octaves + index, noteOctave
```

A secondo dell'ottava, vogliamo dividere **noteFreq** per 1, 2, 4, o 8. Questo significa che l'obbiettivo in realtà è dividere per $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, o $2^3 = 8$. L'istruzione sotto riportata prende il valore di **noteOctave**, che può essere un valore tra 5 ed 8, e lo sottrae da 8. Se **noteOctave** era 8, ora è 0. Se **noteOctave** era 7, ora è 1. Se **noteOctave** era 6, ora è 2, e se **noteOctave** era 5, ora è 3.

```
noteOctave = 8 - noteOctave
```

Ora, **noteOctave** è un valore che può essere usato come esponente di 2, ma come si fa in PBASIC ad elevare 2 ad una potenza? Una risposta è, usare l'operatore **DCD**. **DCD 0** è 1, **DCD 1** è 2, **DCD 2** è 4, e **DCD 3** è 8. Dividere **noteFreq** per **DCD noteOctave** significa che state dividendo per 1, 2, 4, o 8, il che significa che state dividendo **noteFreq** per il giusto valore. Il risultato finale è che **noteFreq** è impostato all'ottava voluta. Userete il Terminale di Debug nella sezione Il Vostro Turno per approfondirne il funzionamento.

```
noteFreq = noteFreq / (DCD noteOctave)
```

Come posso supporre di conoscere l'operatore DCD? Continuate ad imparare ed a fare pratica. Ogni volta che vedete un nuovo, comando, operatore, o qualsiasi altra parola chiave usata negli esempi, cercatela nel Manuale del BASIC Stamp. Informatevi leggendo, e provate ad usarla in programmi di vostra progettazione. Abituatevi alla lettura periodica del Manuale del BASIC Stamp e provate i brevi programmi esempio. Questo è il modo migliore per familiarizzare con i vari comandi ed operatori e come funzionano. Facendo queste cose, svilupperete l'abitudine di aggiungere sempre all'elenco, gli aiuti alla programmazione che potrete usare per la risoluzione dei problemi.

Le prime due righe di codice per la determinazione della durata della nota sono quasi gli stessi del codice del programma esempio precedente. Sebbene, ora, ogni nota può essere punteggiata, il che significa, che la durata può ora essere moltiplicata per 1,5. Viene usato un comando **READ** per leggere il valore memorizzato nella **EEPROM** con la direttiva **DATA Dots**. Viene usata un'istruzione **IF…THEN** per moltiplicare per 3 e dividere per 2 ogni volta che il valore della variabile **noteDot** è 1.

```
READ Durations + index, noteDuration
noteDuration = WholeNote / noteDuration

READ Dots + index, noteDot
IF noteDot = 1 THEN noteDuration = noteDuration * 3 / 2
```

Matematica dei numeri interi. Il BASIC Stamp non processa automaticamente un numero come 1,5. quando esegue calcoli matematici, lavora solo con numeri interi: …, -5, -4, -3, -2, -1, 0, 1, 2, 3, … la miglior soluzione per moltiplicare per 1,5 è di moltiplicare per 3/2. Prima, si moltiplica per 3, e poi si divide per 2.

Ci sono molti modi di programmare il BASIC Stamp per maneggiare valori frazionari. Potete programmare il BASIC Stamp per usare numeri interi che rappresentino la parte frazionaria di un numero. Questo è spiegato nella guida per studenti *Basic Analog and Digital*. Ci sono inoltre due operatori che facilitano il lavoro con i numeri frazionari, ed essi sono: `**` e `*/`. Questi sono spiegati dettagliatamente nella guida per studenti *Applied Sensors* e nel Manuale del BASIC Stamp.

Il resto di questo programma esempio funziona nello stesso modo del precedente Programma di Esempio:

```
    FREQOUT 9, noteDuration, noteFreq

    index = index + 1

  LOOP

  END
```

Il Vostro Turno – Suonare un Motivo con più di un'Ottava.

MusicWithMoreFeatures.bs2 fa uso delle pause, ma rimaneva in un'ottava. Il motivetto "Take Me Out to the Ball Game" mostrato sotto, suona la maggior parte delle sue note nella 6ª ottava. Ci sono due note nella 7ª ottava, e questo fa una grande differenza nel modo in cui suona.

√ Salvate il programma esempio con il nuovo nome MusicWithMoreFeaturesYourTurn.bs2.

√ Modificate il programma sostituendo le quattro direttive data e la dichiarazione di costante con queste:

```
Notes            DATA    "C","C","A","G","E","G","D","P","C","C","A",
                         "G","E","G","Q"
Octaves          DATA      6,  7,  6,  6,  6,  6,  6,  6,  6,  7,  6,
                           6,  6,  6
Durations        DATA      2,  4,  4,  4,  4,  2,  2,  4,  2,  4,  4,
                           4,  4,  2
Dots             DATA      0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,
                           0,  0,  1
```

```
BeatsPerMin      CON     240
```

√ Lanciate il programma e verificate che suoni correttamente.

Quelle due note nella 7ª ottava sono essenziali per far suonare correttamente il motivetto. È interessante ascoltare che cosa accade se quei due valori 7 sono cambiati in 6.

√ Provate a cambiare i due valori 7 nella direttiva **DATA Octaves** in modo che siano 6. Ricordatevi, questo farà suonare in maniera strana la canzone "Take Me out to the Ball Game".
√ Lanciate il programma, ed ascoltate l'effetto delle ottave sbagliate sulla canzone
√ Rimettetete **DATA Octaves** nel suo stato originale.
√ Lanciate di nuovo il programma ed ascoltate se ora suona correttamente.

ESERCIZIO #5: SUONERIE DI TELEFONI CELLULARI

Molti telefoni cellulari suonano musica che può essere scaricata dalle pagine web. Il computer invia i dati delle note al microcontrollore nel telefono, il quale suona le note ogni volta che arriva una chiamata. Queste sono chiamate suonerie di avviso o spesso semplicemente suonerie.

Uno dei sistemi più largamente usati per comporre, registrare ed inviare note è uno che implica la scrittura di stringhe di testo che descrivono ciascuna nota nella canzone. Di seguito c'è un esempio di come sono le prime note della 5ª di Beethoven in formato RTTTL:

Beethoven5:d=8,o=7,b=125:g,g,g,2d#,p,f,f,f,2d

Questo formato per la memorizzazione dei dati musicali è chiamato RTTTL, che sta per l'inglese Ringing Tone Text Transfer Language (Linguaggio per il Trasferimento in Testo dei Toni di Chiamata). La cosa grande dei file RTTTL è che sono largamente usati nel World Wide Web. Molti siti hanno file RTTTL a disposizione per il download gratuito. Ci sono anche programmi software gratuiti che possono essere usati per comporre od emulare questi file così come per scaricarli nel vostro cellulare. Anche le specifiche RTTTL sono pubblicate nel World Wide Web. L'Appendice G riassume come un file RTTTL memorizza le note, le durate, le pause, il tempo e le note punteggiate.

Questo Esercizio spiega alcune tecniche di programmazione PBASIC che possono essere usate per riconoscere differenti elementi di testo. La capacità di riconoscere caratteri o

gruppi di caratteri differenti ed eseguire azioni basate sul contenuto di quei caratteri è estremamente utile. Infatti, è la chiave per la conversione del formato per suonerie RTTTL (come la 5 [a] di Beethoven sopra) in musica. Alla fine di questo Esercizio, c'è un programma applicativo che potete usare per suonare toni di chiamata in formato RTTTL.

Selezionare quale Blocco di Codice Eseguire Caso per Caso

L'istruzione **SELECT…CASE** è probabilmente l'attrezzo migliore per il riconoscimento dei caratteri o dei valori. Ricordate che questo è uno degli attrezzi usati per convertire un tono di chiamata RTTTL in note musicali. In generale, **SELECT…CASE** è usato per:

- Selezionare una variabile o espressione.
- Valutare quella variabile o espressione caso per caso.
- Eseguire del codice secondo in quale caso il valore di quella variabile è appropriato.

8

Questa è la sintassi di **SELECT…CASE**:

```
SELECT expression
  CASE condition(s)
  statement(s)
ENDSELECT
```

Per vedere come funziona **SELECT…CASE** potete provare i prossimi due programmi esempio. SelectCaseWithValues.bs2 prende il valore numerico che voi inserite nel Terminale di Debug ed in risposta vi dice la grandezza minima di variabile che dovrete usare per contenere quel valore. SelectCaseWithCharacters.bs2 vi dice se il carattere che avete digitato nel Terminale di Debug è maiuscolo o minuscolo, un numero o un segno di punteggiatura.

Ricordate che per trasmettere i caratteri digitati al BASIC Stamp dovete usare la finestra superiore del Terminale di Debug. Le finestre di trasmissione e di ricezione sono mostrate in Figura 8-5.

Finestre

Trasmissione →

Ricezione →

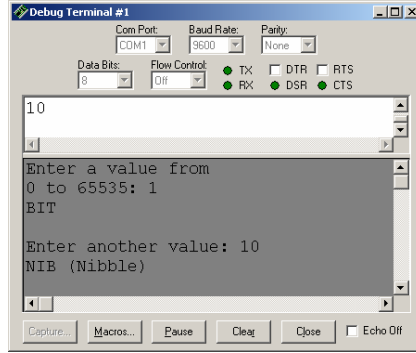



Figura 8-5
Invio dei messaggi al BASIC Stamp

Clickate nella finestra di trasmissione (superiore) ed inserite il valore od i caratteri che volete trasmettere al BASIC Stamp.

Programma di Esempio: SelectCaseWithValues.bs2

√ Digitate e lanciateSelectCaseWithValues.bs2.
√ Clickate nella finestra di trasmissione del Terminale di Debug.
√ Digitate un valore tra 0 e 65535. Ricordate di premere il tasto Enter per inviare il vostro valore.
√ La Tabella 2-2 (sotto) è una copia della tabella a pagina 57. Usate questa tabella per verificare che il programma esempio prenda le decisioni corrette circa la grandezza dei numeri che digitate nel Terminale di Debug.

Tabella 2-2: Tipi di Variabile e Valori che possono contenere

Tipo di Variabile	Gamma dei Valori
Bit	0 to 1
Nib	0 to 15
Byte	0 to 255
Word	0 to 65535

Che cosa accade se inserite un numero maggiore di 65535? Se inserite il numero 65536, il BASIC Stamp memorizzerà il numero 0. Se inserite il numero 65537, il BASIC Stamp memorizzerà il numero 1, e così via. Quando un numero è troppo grande per la variabile che lo deve contenere, si chiama overflow.

```
' Che cosa è un Microcontrollore- SelectCaseWithValues.bs2
' Digitate un valore e vedete la variabile necessaria a contenerlo.

'{$STAMP BS2}
'{$PBASIC 2.5}
```

```
value          VAR     Word

DEBUG "Digitate un valore da", CR,
      "0 a 65535: "

DO

  DEBUGIN DEC value

  SELECT value

    CASE 0, 1
      DEBUG "Bit", CR
      PAUSE 100

    CASE 2 TO 15
      DEBUG "Nib (Nibble)", CR
      PAUSE 200

    CASE 16 TO 255
      DEBUG "Byte", CR
      PAUSE 300

    CASE 256 TO 65535
      DEBUG "Word", CR
      PAUSE 400

  ENDSELECT

  DEBUG CR, "Digitate un altro valore: "

LOOP
```


Come funziona SelectCaseWithValues.bs2

Per inserire i valori nel Terminale di Debug viene dichiarata una variabile word.

```
value VAR Word
```

Il comando **`DEBUGIN`** prende il numero che avete digitato e lo mette nella variabile **`value`**.

```
DEBUGIN DEC value
```

La dichiarazione **`SELECT`** sceglie la variabile **`value`** per effettuare la valutazione.

```
SELECT value
```

Il primo caso è se la variabile **`value`** è uguale a 0 o ad 1. Se **`value`** è uguale a qualsiasi dei due vengono eseguiti i comandi che lo seguono, **`DEBUG`** e **`PAUSE`**.

```
CASE 0, 1
```

```
      DEBUG "BIT", CR
      PAUSE 100
```

Il secondo caso è se **`value`** è uguale a qualsiasi numero da 2 a 15. Se è uguale ad uno di questi numeri, vengono eseguiti i comandi **`DEBUG`** e **`PAUSE`** sotto di lui.

```
    CASE 2 to 15
      DEBUG "NIB (Nibble)", CR
      PAUSE 200
```

Quando tutti i casi sono stati esaminati, viene usata la parola chiave **`ENDSELECT`** per completare l'istruzione **`SELECT..CASE`**.

```
  ENDSELECT
```

Programma di Esempio: SelectCaseWithCharacters.bs2

Questo programma esempio valuta ciascun carattere che digitate nella finestra di trasmissione del Terminale di Debug. Può riconoscere caratteri maiuscoli, minuscoli, cifre e qualche segno di punteggiatura. Se inserite un carattere che il programma non riconosce, vi dirà di provare di nuovo (digitando un'altro carattere).

- √ Digitate e lanciateSelectCaseWithCharacters.bs2.
- √ Clickate nella finestra di trasmissione del Terminale di Debug, digitate alcuni caratteri ed osservate i risultati.

```
' Che cosa è un Microcontrollore- SelectCaseWithCharacters.bs2
' Programma che può identificare alcuni caratteri:
' Maiuscole/minuscole, cifre, punteggiatura.

'{$STAMP BS2}
'{$PBASIC 2.5}

character        VAR     Byte

DEBUG "Digitate un Carattere: ", CR

DO

  DEBUGIN character

  SELECT character

    CASE "A" TO "Z"
      DEBUG CR, "Maiuscola", CR

    CASE "a" TO "z"
```

```
      DEBUG CR, "Minuscola", CR

    CASE "0" TO "9"
      DEBUG CR, "Numero", CR

    CASE "!", "?", ".", ","
      DEBUG CR, "Segno di Punteggiatura", CR

    CASE ELSE
      DEBUG CR, "Carattere Sconosciuto.", CR,
                "Provate con un carattere diverso."

  ENDSELECT

  DEBUG CR, "Digitate un’altro Carattere", CR

LOOP
```


Come funziona SelectCaseWithCharacters.bs2

Questo programma esempio, se confrontato con SelectCaseWithValues.bs2, è leggermente diverso. Primo, il nome della variabile **`value`** è stato cambiato in **`character`**, Secondo la sua dimensione è variata da word a byte. Perché tutti i caratteri in PBASIC hanno dimensione byte. L’istruzione **`SELECT`** sceglie la variabile **`character`** per una valutazione caso per caso.

```
SELECT character
```

Le virgolette sono usate per indicare all’Editor del BASIC Stamp che vi state riferendo ad un carattere.

```
SELECT character

  CASE "A" to "Z"
    DEBUG CR, "Maiuscola", CR

  CASE "a" to "z"
    DEBUG CR, "Minuscola", CR

  CASE "0" to "9"
    DEBUG CR, "Numero", CR

  CASE "!", "?", ".", ","
    DEBUG CR, "Punteggiatura", CR
```

C’è anche una diversa istruzione **`CASE`** che non era stata usata nell’esempio precedente:

```
  CASE ELSE
```

```
            DEBUG CR, "Carattere Sconosciuto.", CR,
                      "Provate con un carattere diverso."
```

Questa istruzione **CASE** dice al blocco di codice **SELECT** che cosa fare se nessuno degli altri casi è vero. Potete incorrere in questa evenienza digitando un carattere come ad esempio % o $.

Il Vostro Turno – Selezione Basata su una Gamma

√ Modificate l'istruzione **SELECT…CASE** in SelectCaseWithCharacters.bs2 in maniera che visualizzi "Caratteri Speciali" quando digitate uno dei seguenti caratteri: @, #, $, %, '^' , &, *, (,), _, o +.

Programma Applicativo per suonare toni di suoneria RTTTL.

Viene di seguito descritto il file RTTTL che contiene le informazioni musicali usate nel prossimo programma esempio. Nella sezione Il Vostro Turno ci sono altre cinque direttive **DATA RTTTL_File** che potete provare. Questo programma suona un motivo chiamato Sveglia, che viene suonato al mattino nelle caserme. Potete averlo sentito in molti film al cinema od alla televisione.

```
RTTTL_File     DATA    "Sveglia:d=4,o=7,b=140:8g6,8c,16e,16c,8g6,8e,",
                       "8c,16e,16c,8g6,8e,8c,16e,16c,8a6,8c,e,8c,8g6,",
                       "8c,16e,16c,8g6,8e,8c,16e,16c,8g6,8e,8c,16e,",
                       "16c,8g6,8e,c,p,8e,8e,8e,8e,g,8e,8c,8e,8c,8e,8c,",
                       "e,8c,8e,8e,8e,8e,8e,g,8e,8c,8e,8c,8g6,8g6,c."
```

Programma di Esempio: MicroMusicWithRtttl.bs2

Questo programma applicativo è abbastanza lungo, ed è una buona idea scaricare l'ultima versione dalla pagina www.parallax.com → Downloads → Educational Curriculum. Cercate un link chiamato *Selected Example Programs*. Scaricarlo ed aprirlo con l'Editor del BASIC Stamp dovrebbe farvi risparmiare parecchio tempo. L'alternativa, certamente, è digitare e controllare quattro pagine di codice.

√ Usate l'Editor del BASIC Stamp per aprire il file appena scaricato MicroMusicWithRtttl.bs2, o digitate molto attentamente l'esempio sotto riportato.

√ Lanciate il programma, e verificate che il motivo della sveglia sia chiaramente riconoscibile.

√ Andate alla sezione Il Vostro Turno e provate qualche altro motivo (RTTTL_File DATA directives).

```
' Che cosa è un Microcontrollore- MicroMusicWithRtttl.bs2
' Suona le suonerie in formato Nokia RTTTL usando la direttiva DATA.

'{$STAMP BS2}
'{$PBASIC 2.5}

' -----[ I/O Definitions ]-------------------------------------------------

  SpeakerPin     CON     9    ' Altoparlante Piezo collegato a P9.

' -----[ Variables ]-------------------------------------------------------

  counter        VAR     Word ' Contatore di uso generico.
  char           VAR     Byte ' Variabile per la memorizzazione dei caratteri.
  index          VAR     Word ' Indice per puntare a data.

  noteLetter     VAR     Byte ' Memorizza il carattere della nota.
  noteFreq       VAR     Word ' Memorizza la frequenza della nota.
  noteOctave     VAR     Word ' Memorizza l'ottava della nota.

  duration       VAR     Word ' Memorizza la durata della nota.
  tempo          VAR     Word ' Memorizza il tempo.

  default_d      VAR     Byte ' Memorizza la durata di default.
  default_o      VAR     Byte ' Memorizza l'ottava di default.
  default_b      VAR     Word ' Memorizza il default delle battute/minuto.

' -----[ EEPROM Data ]-----------------------------------------------------

  RTTTL_File     DATA    "Sveglia:d=4,o=7,b=140:8g6,8c,16e,16c,8g6,8e,",
                         "8c,16e,16c,8g6,8e,8c,16e,16c,8a6,8c,e,8c,8g6,",
                         "8c,16e,16c,8g6,8e,8c,16e,16c,8g6,8e,8c,16e,",
                         "16c,8g6,8e,c,p,8e,8e,8e,8e,g,8e,8c,8e,8c,8e,8c,",
                         "e,8c,8e,8e,8e,8e,8e,g,8e,8c,8e,8c,8g6,8g6,c."

  Done           DATA    ",q,"

  Notes          DATA     "p",       "a",       "#",       "b",
                          "c",       "#",       "d",       "#",
                          "e",       "f",       "#",       "g",
                          "#"
  Octave8        DATA    Word 0,    Word 3520, Word 3729, Word 3951,
                         Word 4186, Word 4435, Word 4699, Word 4978,
                         Word 5274, Word 5588, Word 5920, Word 6272,
                         Word 6645

' -----[ Initialization ]--------------------------------------------------

  counter = 0                              ' Initializza il contatore.
```

8

```
  GOSUB FindEquals                           ' Trova il primo '=' nel file.
  GOSUB ProcessDuration                      ' Prende la durata di default.
  GOSUB FindEquals                           ' Trova il prossimo '='.
  GOSUB ProcessOctave                        ' Prende l'ottava di default.
  GOSUB FindEquals                           ' Trova l'ultimo '='.
  GOSUB GetTempo                             ' Prende il tempo di default.

' -----[ Program Code ]--------------------------------------------------

  DO UNTIL char = "q"                        ' Ciclo fino alla 'q' in DATA.
    GOSUB ProcessDuration                    ' Prende la durata della nota.
    GOSUB ProcessNote               ' Prende il valore dell'indice della nota.
    GOSUB CheckForDot                        ' Se punteggiata, durata 3/2.
    GOSUB ProcessOctave                      ' Prende l'ottava.
    GOSUB PlayNote                 ' Prende la frequenza, suona la nota, next.
  LOOP                                       ' Fine del ciclo principale.

  END                                        ' Fine del programma.

' -----[ Subroutine - Find Equals Character ]----------------------------

  FindEquals:                                ' Cerca '=' nei caratteri del
                                             ' file RTTTL.
    DO                                       ' Incrementa il contatore fino a
      READ RTTTL_File + counter, char        ' che trova '=', quindi
      counter = counter + 1                  ' ritorna.
    LOOP UNTIL char = "="

    RETURN

' -----[ Subroutine - Read Tempo from RTTTL Header ]----------------------

  GetTempo:                                  ' Analizza il file RTTTL per
                                             ' trovare il Tempo.
                                             ' Converte i caratteri in numeri
    default_b = 0                            ' sottraendo 48 dal valore ASCII
    DO                                       ' di ciascun carattere.
      READ RTTTL_File + counter, char        ' Iterativamente moltiplica
      IF char = ":" THEN                     ' ciascuna cifra per 10 se c'è
        default_b = default_b / 10           ' un'altra cifra, quindi aggiunge
        counter = counter + 1                ' la cifra più recente alla
        EXIT                                 ' colonna delle unità.
      ENDIF                                  ' Per esempio, la stringa
        default_b = default_b + char - 48    ' "120" è (1 X 10 X 10)
        counter = counter + 1                ' + (2 X 10) + 0. Prima viene
        default_b = default_b * 10           ' convertito '1', quindi viene
    LOOP UNTIL char = ":"                    ' moltiplicato per 10. Il '2' è
                                             ' quindi convertito/addizionato.
    RETURN                                   ' 0 è convertito/addizionato.

' -----[ Subroutine - Look up Octave ]-----------------------------------
```

```
  ProcessOctave:                            ' L'ottava può o essere o no
                                            ' inclusa in una data nota
    READ RTTTL_File + counter, char         ' perché ciascuna nota che è
    SELECT char                             ' suonata nell'ottava di default
      CASE "5" TO "8"                       ' non specifica l'ottava.
        noteOctave = char - "0"             ' Se un carattere è da '5' a '8'
        counter = counter + 1               ' la usa, altrimenti
      CASE ELSE                             ' usa default_o.
        noteOctave = default_o              ' I caratteri sono convertiti
    ENDSELECT                               ' In numeri sottraendo '0', che
    IF default_o = 0 THEN                   ' è lo stesso che sottrarre 48.
      default_o = noteOctave                ' La prima volta  che questa
    ENDIF                                   ' subroutine viene chiamata,
                                            ' default_o è 0. E se è 0, allora
    RETURN                                  ' default_o viene impostato a 0.

' -----[ Subroutine - Find Index of Note ]-------------------------------

  ProcessNote:                              ' Imposta il valore di index del
                                            ' Lookup della frequenza del
    READ RTTTL_File + counter, char         ' carattere della nota. Se 'p',
    SELECT char                             ' index è 0. Se è da 'a' a 'g',
      CASE "p"                              ' legge il valore del carattere
        index = 0              ' nella tabella DATA e trova la corrispondenza.
        counter = counter + 1  ' Quando è trovata la corrispondenza
      CASE "a" TO "g"         ' registra il valore dell'indice. Se il prossimo
        FOR index = 1 TO 12    ' carattere è un diesis (#), aggiunge 1 al
          READ Notes + index, noteLetter    ' valore dell'indice per
          IF noteLetter = char THEN EXIT    ' aumentare l'indice (e la
        NEXT                                ' frequenza) di un 1 semitono.
        counter = counter + 1               ' Come con le altre subroutine,
        READ RTTTL_File + counter, char     ' incrementa counter per ogni
        SELECT char                         ' carattere che è processato.
          CASE "#"
            index = index + 1
            counter = counter + 1
        ENDSELECT
    ENDSELECT

    RETURN

' -----[ Subroutine - Determina La durata della Nota ]------------------------

  ProcessDuration:                         ' Controlla per vedere se i carat-
                                            ' formano 1, 2, 4, 8, 16 o 32.
    READ RTTTL_File + counter, char         ' Se sì, converte dal carattere
                                            ' ASCII ad un valore sottraendo
    SELECT char                             ' 48. Nel caso di 16 o 32,
      CASE "1", "2", "3", "4", "8"          ' moltiplica per 10 ed aggiunge
        duration = char - 48                ' la prossima cifra alla colonna
```

```
        counter = counter + 1                ' delle unità.
        READ RTTTL_File + counter, char
        SELECT char
          CASE "6", "2"
            duration = duration * 10 + char - 48
            counter = counter + 1
        ENDSELECT
        CASE ELSE                            ' Se è assente la durata, usa
        duration = default_d                 ' il valore di default.
    ENDSELECT

    IF default_d <> 0 THEN                   ' Se default_d non è definito
      duration = 60000/default_b/duration*3  ' (if default_d = 0), allora
    ELSE                                     ' imposta la durata di
      default_d = duration                   ' default_d = usando d=#.
    ENDIF

    RETURN

' -----[ Subroutine - Check For '.' Indicating 1.5 Duration ]-------------

  CheckForDot:                               ' Controlla la punteggiatura che
                                             ' moltiplica la durata per 3/2.
    READ RTTTL_File + counter, char          ' Se esiste, moltiplica per
    SELECT char                              ' 3/2 ed incrementa il contatore,
      CASE "."                               ' altrimenti, non fare niente
        duration = duration * 3 / 2          ' e ritorna.
        counter = counter + 1
    ENDSELECT

    RETURN

' -----[ Subroutine - Find Comma and Play Note/Duration ]-----------------

  PlayNote:                                  ' Trova l'ultima virgola nella
                                             ' nota corrente. Quindi, estrai
    READ RTTTL_File + counter, char          ' da data la frequenza della
    SELECT char                              ' nota, e suonala, oppure se la
      CASE ","                               ' frequenza è = 0 fai una pausa.
        counter = counter + 1
        READ Octave8 + (index * 2), Word noteFreq
        noteOctave = 8 - noteOctave
        noteFreq = noteFreq / (DCD noteOctave)
        IF noteFreq = 0 THEN
           PAUSE duration
        ELSE
          FREQOUT SpeakerPin, duration, noteFreq
        ENDIF
    ENDSELECT

    RETURN
```

Come funziona MicroMusicWithRtttl.bs2

Questo programma esempio è divertente da usare, ed illustra il genere di codice che con un poco di pratica sarete in grado di scrivere. Sebbene, sia stato incluso in questo testo più per divertimento che per i concetti di codifica che impiega.

Se esaminate brevemente il codice, potrete notare che avete già usato tutti i comandi e gli operatori presenti nel programma. Viene di seguito elencata una lista degli elementi di questa applicazione che, dovrebbero ormai, esservi familiari:

- Commenti che vi aiutano a spiegare il codice
- Dichiarazione delle Costanti e delle variabili
- Dichiarazioni **DATA**
- Comandi **READ**
- Blocchi **IF…ELSE…ENDIF**
- **DO…LOOP** sia con che senza **WHILE** ed **UNTIL**
- Subroutine con **GOSUB**, etichette, e **RETURN**
- Cicli **FOR…NEXT**
- Comandi **LOOKUP** e **LOOKDOWN**
- I comandi **FREQOUT** e **PAUSE**
- **SELECT…CASE**

8

Il Vostro Turno – Motivi diversi

√ Provate a sostituire la direttiva **DATA RTTTL_File** nel programma esempio MicroMusicWithRTTTL.bs2 con ciascuno dei cinque diversi file musicali sotto elencati.

Solamente una direttiva DATA RTTTL_File alla volta directive! Assicuratevi di sostituire, e non aggiungere, la nuova direttiva **DATA RTTTL_File**.

√ Lanciate MicroMusicWithRTTTL.bs2 per provare ciascun file RTTTL.

```
RTTTL_File    DATA    "TwinkleTwinkle:d=4,o=7,b=120:c,c,g,g,a,a,2g,f,",
                      "f,e,e,d,d,2c,g,g,f,f,e,e,2d,g,g,f,f,e,e,2d,c,c,",
                      "g,g,a,a,2g,f,f,e,e,d,d,1c"

RTTTL_File    DATA    "FrereJacques:d=4,o=7,b=125:c,d,e,c,c,d,e,c,e,f",
                      ",2g,e,f,2g,8g,8a,8g,8f,e,c,8g,8a,8g,8f,e,c,c,g6",
                      ",2c,c,g6,2c"
```

```
RTTTL_File    DATA    "Beethoven5:d=8,o=7,b=125:g,g,g,2d#,p,f,f,f,2d"

RTTTL_File    DATA    "ForHe'sAJollyGoodFellow:d=4,o=7,b=320:c,2e,e,e,",
                      "d,e,2f.,2e,e,2d,d,d,c,d,2e.,2c,d,2e,e,e,d,e,2f,",
                      "g,2a,a,g,g,g,2f,d,2c"

RTTTL_File    DATA    "TakeMeOutToTheBallgame:d=4,o=7,b=225:2c6,c,a6,",
                      "g6,e6,2g.6,2d6,p,2c6,c,a6,g6,e6,2g.6,g6,p,p,a6",
                      ",g#6,a6,e6,f6,g6,a6,p,f6,2d6,p,2a6,a6,a6,b6,c,",
                      "d,b6,a6,g6"
```

Scaricare file RTTTL: Come menzionato precedentemente, in vari siti del World Wide Web ci sono moltissimi file RTTTL disponibili per essere scaricati. Questi file sono messi a disposizione da entusiasti delle suonerie, molti dei quali non sono esperti di musica. Alcune suonerie sono molto ben fatte, altre sono appena riconoscibili.

Se volete scaricare e suonare qualche altro file RTTTL, assicuratevi di togliere tutti gli spazi tra I caratteri, quindi inserite il file di testo fra virgolette.

SOMMARIO

Questo Capitolo ha presentato le tecniche per creare suoni e toni musicali usando il BASIC Stamp ed un altoparlante piezoelettrico. Il comando **FREQOUT** può essere usato per inviare segnali on/off all'Altoparlante piezoelettrico in modo che emetta effetti sonori e/o note musicali. Il comando **FREQOUT** ha argomenti che controllano il Pin I/O al quale il segnale viene inviato, la ***Duration*** del tono, la frequenza del tono (***Freq1***). L'argomento opzionale ***Freq2*** può essere usato per miscelare toni.

Gli effetti sonori possono essere ottenuti regolando la frequenza e la durata dei toni e le pause intermedie. Il valore della frequenza può inoltre essere variato in una gamma di valori o miscelato per creare una varietà di effetti.

Anche la creazione di note musicali dipende da frequenza, durata e pause. Il valore dell'argomento ***Duration*** del **FREQOUT** comando è determinato dal Tempo della canzone e la durata della nota (quattro quarti, due quarti, un quarto, etc.). il valore ***Freq1*** della nota è determinato dalla lettera che identifica la nota e dall'ottava. Le pause tra le note sono usate per impostare la durata del comando **PAUSE**.

8

Suonare semplici canzoni con usando il BASIC Stamp può essere fatto con una sequenza di comandi **FREQOUT**, ma ci sono modi migliori per memorizzare e recuperare dati musicali. Le direttive **DATA** insieme con le loro etichette opzionali ***Symbol*** sono state usate per memorizzare valori byte senza prefisso e valori word con il prefisso **Word**. Il comando **READ** è stato usato per recuperare valori memorizzati dalle direttive **DATA**. L'argomento ***Address*** del comando **READ** usa sempre l'etichetta ***Symbol*** opzionale della direttiva **DATA** per distinguere fra diversi tipi di dati. Alcune etichette dei simboli che sono state usate erano **Notes**, **Durations**, **Dots**, ed **Octaves**.

I dati musicali possono essere memorizzati in formati che derivano dalla notazione usata negli spartiti. Lo stile dei dati degli spartiti può quindi essere convertito in **Frequency** usando i comandi **LOOKUP** e **LOOKDOWN**. Sui valori delle variabili possono anche essere effettuate operazioni matematiche per cambiare l'ottava di una nota dividendo la sua frequenza per una potenza di due. Le operazioni matematiche sono utili anche per la durata delle note a partire dal Tempo o dalla durata della nota intera.

È stato introdotto **SELECT…CASE** come modo di valutare una variabile caso per caso. **SELECT…CASE** è particolarmente utile nell'esame di caratteri o numeri quando ci siano da

fare molte scelte circa una variabile o quando ci siano diversi insiemi di azioni che debbono essere eseguite in base al valore di una variabile. È stato usato un programma, che converte stringhe di caratteri deh descrivono toni musicali per telefoni cellulari (chiamati file RTTTL), per presentare un programma più grande che fa uso di tutte le tecniche di programmazione introdotte in questo testo. In questo programma **SELECT…CASE** ha giocato un ruolo predominante perché viene usato per esaminare caratteri selezionati caso per caso in un file RTTTL.

Domande

1. Che cosa fa suonare un tono in modo acuto? Che cosa fa suonare un tono in modo grave?
2. Che cosa fa **FREQOUT 15, 1000, 3000**? Quale effetto ha ciascuno dei numeri?
3. Quali sono i tre elementi principali degli effetti sonori?
4. Potete inserire un ciclo **FOR…NEXT** all'interno di un altro? Come funziona?
5. Come potete usare il commando **FREQOUT** della seconda domanda per fargli emettere due frequenze contemporaneamente?
6. Se battete il tasto B6 di un piano, quale frequenza emette?
7. Come usate il comando **FREQOUT** per emettere note musicali?
8. Dove vengono memorizzati i dati della direttiva **DATA**?
9. Quanta memoria occorre per memorizzare un carattere?
10. Quanta memoria è una word?
11. Come modifichereste una direttiva **DATA** o un commando **READ** se voleste memorizzare e recuperare dei valori word?
12. Potete avere più di una direttiva **DATA**? Se si, come indichereste ad un comando **READ** di prendere i dati dall'una o dall'altra direttiva **DATA**?
13. Domanda intrigante: Quanto dura un quarto di nota?
14. Quali comandi potete usare per tradurre le lettere delle note (caratteri) di una direttiva **DATA** nelle loro rispettive frequenze?
15. Che cos'è un'ottava? Se conoscete la frequenza di una nota in un ottava, che cosa dovete fare con quella frequenza per suonare quella nota nell'ottava successiva?
16. Che cos'è il Tempo? Come calcolate il valore di un quarto a partire dal Tempo? Come trovate il valore di quattro quarti?
17. Che cosa fa l'operatore **DCD**? Che cosa significa **DCD 3**? e **DCD 4**? e **DCD 5**?
18. Che cosa fa **SELECT…CASE**?

Esercizi

1. Modificate il tono di "Alarm…" di ActionTones.bs2 in modo che la frequenza del tono che viene suonato aumenti di 500 ogni volta che il tono viene ripetuto.
2. Modificate NestedLoops.bs2 in modo che anche l'argomento ***Freq2*** del comando **FREQOUT** usi il valore della variabile **duration**.
3. Modificate DoReMiFaSolLaTiDo.bs2 in modo che suoni tutte le note dell'ottava da C5 fino a C6.
4. Modificate i comandi **DEBUG** in TwinkleTwinkle.bs2 in modo che visualizzi la durata di ciascuna nota sia in quarti (una durata di 500) o due quarti (una durata di 1000).
5. Modificate NotesAndDurations.bs2 in modo che calcoli il valore di una variabile chiamata **wholeNote** in base al valore di una costante chiamata **Tempo**.
6. Spiegare come modifichereste MusicWithMoreFeatures.bs2 in modo che suoni per la stessa nota due frequenze allo stesso tempo.
7. Spiegare come modificare MusicWithMoreFeatures.bs2 in modo che visualizzi un messaggio di avvertimento nel Terminale di Debug ogni volta che viene suonata una nota punteggiata.
8. Spiegate che cosa fa la subroutine **CheckForDot** in MicroMusicWithRtttl.bs2 In che modo converte una nota in una nota punteggiata?

8

Progetti

1. Costruite il generatore controllato da due tasti. Se viene premuto un tasto, l'altoparlante dovrebbe emettere un bip di 2 kHz per 1/5 di secondo. Se viene premuto l'altro tasto l'altoparlante dovrebbe emettere un bip di 3 kHz per 1/10 di secondo.
2. Modificate il programma del primo progetto in modo che ad ogni pressione e rilascio di uno dei tasti, suoni una nota musicale per ½ secondo. Se premete un tasto, la nota dovrà aumentare di un tono. Se premete l'altro tasto, dovrà diminuire di un tono.
3. Per controllare il tempo di TwinkleTwinkle.bs2 usate un circuito con Potenziometro.
4. Creare un'applicazione che generi toni in base alla luminosità di una stanza. Per questa applicazione vi servirà un circuito con fotoresistenza. L'altoparlante dovrebbe emettere un bip al secondo. Ed il tono dovrebbe diventare più acuto all'aumentare della luminosità e più grave al diminuire della luminosità.
5. Create un generatore musicale. Usate un Potenziometro per selezionare le note, ed il tasto per suonarle. Opzionalmente, usate due tasti. Se tutti e due i tasti sono

premuti, suona due quarti. Se è premuto un tasto, suona un quarto, e se è premuto l'altro tasto, suona un ottavo.

6. Costruite un carillon con due tasti, un display LED a 7 segmenti ed un altoparlante piezo. Questo carillon suona uno di tre motive che vengono selezionati usando i tasti ed il display LED a 7 segmenti. Un tasto può essere usato per scorrere i numeri da 1 a 3 sul display LED a 7 segmenti. L'altro tasto suona la canzone.

Ulteriori Approfondimenti

"*Applied Sensors*", Student Guide, Version 2.0, Parallax Inc., 2003

Sono spiegati altri effetti sonori, click e scricchiolii che usano l'altoparlante piezoelettrico. La scala pentatonica e la scala bentemperata sono alla base della discussione della matematica frazionale. L'altoparlante viene usato anche come avvisatore per una varietà di misure con sensori.

"*Basic Analog and Digital*", Student Guide, Version 2.0, Parallax Inc., 2003

Viene usato l'altoparlante per rendere udibile una frequenza generate da un dispositivo chiamato temporizzatore 555. Il BASIC Stamp è usato per misurare la frequenza del tono con un comando che si chiama **COUNT**.

"*Understanding Signals*", Student Guide, Version 1.0, Parallax Inc., 2003

Potete usare le informazioni di questo libro per visualizzare l'uscita del comando **FREQOUT,** sia come impulso digitale, che come onda sinusoidale.

Capitolo #9: Blocchi di Costruzione Elettronica

QUELLE COSINE NERE

Non dovete far altro che guardare il vostro BASIC Stamp (vedere la Figura 9-1) per trovare esempi di "quelle cosine nere" (circuiti integrati) in gergo chip. Ciascuno di questi ha una funzione speciale. Il chip in alto a destra è il regolatore di tensione. Questo chip porta la tensione della pila ad esattamente 5.0 V, che sono necessari per il corretto funzionamento del resto dei componenti del BASIC Stamp. Il chip in alto a sinistra è la EEPROM del BASIC Stamp. I programmi PBASIC sono convertiti in numeri chiamati token che sono scaricati nel BASIC Stamp. Questi token sono memorizzati nella EEPROM, e li potete vedere nell'Editor del BASIC Stamp clickando *Run* e poi *Memory Map*. Il chip più grande si chiama chip Interprete. Estrae i token dalla EEPROM ed interpreta i comandi PBASIC che i token rappresentano. Quindi, esegue il commando, estrae il prossimo token, e così via. Questo processo si chiama "estrazione ed esecuzione".

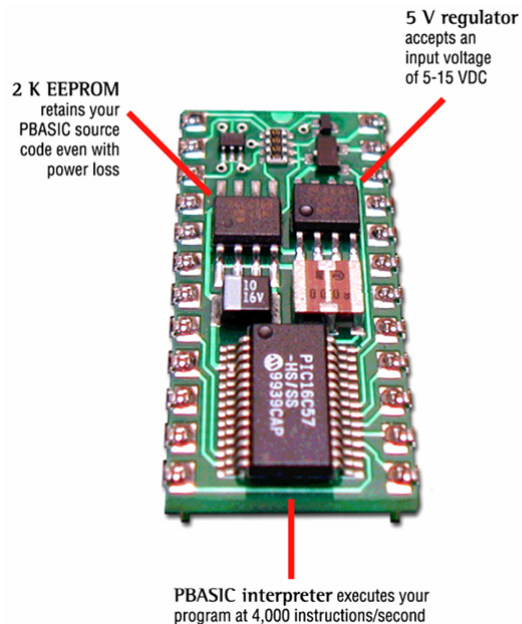



Figura 9-1
Circuiti Integrati sul BASIC Stamp

9

La gente usa il termine "circuito integrato" (IC) per indicare "le cosine nere". Un circuito integrato in realtà un sottile pezzettino di silicio all'interno di un contenitore di plastica nera o di ceramica. A secondo del chip, può contenere tra le centinaia ed i milioni di transistor. Il transistor è il blocco base dei circuiti integrati, ed in questo capitolo avrete l'opportunità di sperimentare con un transistor. Altri componenti familiari che sono inseriti nei circuiti integrati includono i diodi, le resistenze ed i condensatori.

Prendetevi una pausa per ripensare agli esercizi di questo libro che avete fatto fino ad ora. L'elenco include accendere e spengere LED, leggere lo stato dei tasti, controllare servomotori, leggere potenziometri, misurare luci, controllare display e creare suoni. Sebbene questo non sia che l'inizio, è già notevole, specialmente considerando che potete combinare questi esercizi per creare giochi e gagdet. Il nucleo del sistema che rende possibili tutte queste attività è racchiuso nei tre circuiti integrati mostrati nella Figura 9-1 ed in alcuni componenti aggiuntivi. Questo vi dimostra quanto possono essere potenti i circuiti integrati quando sono progettati per lavorare insieme.

ESPANDETE I VOSTRI PROGETTI CON CIRCUITI INTEGRATI PERIFERICI

Ci sono migliaia di circuiti integrati progettati per essere usati con i microcontrollori. Qualche volta diversi costruttori di circuiti integrati producono chip che hanno la stessa funzione. Qualche volta le caratteristiche di ciascun chip differiscono leggermente, ed altre volte, i chip sono identici, ma uno può costare un poco meno dell'altro. Ciascuno delle migliaia di diversi circuiti integrati può essere usato come "mattone per una varietà di progetti. Le fabbriche pubblicano le informazioni su come funzionano i loro circuiti integrati in documenti chiamati datasheet che sono pubblicati sul World Wide Web. Questi fabbricanti pubblicano anche le note applicative, che mostrano come usare i loro circuiti integrati in modi utili o univoci, che facilitano la progettazione dei prodotti. I produttori di circuiti integrati forniscono queste informazioni nella speranza che i progettisti le useranno per inserire i loro chip nel giocattolo o nell'applicazione all'ultimo grido. Se vengono venduti migliaia di giocattoli, significa che le fabbriche vendono migliaia dei loro circuiti integrati.
In questo capitolo, esperimenterete con un transistor ed un circuito integrato speciale chiamato Potenziometro digitale. Come menzionato precedentemente, il transistor è il mattone dei circuiti integrati. Come è anche il mattone per la costruzione di molti altri circuiti. Il Potenziometro digitale, inoltre ha una grande varietà di usi. Ricordatevi comunque che per ogni esercizio che avete fatto, ci sono probabilmente centinaia di maniere diverse in cui avreste potuto usare questi circuiti integrati.

ESERCIZIO #1: CONTROLLARE IL FLUSSO DI CURRENTE CON UN TRANSISTOR

In questo Esercizio, userete un transistor come modo per regolare la corrente che scorre in un LED. Potete usare il LED, per tenere sotto controllo la corrente, dal momento che brilla di più quando la corrente è maggiore e di meno quando la corrente è minore.

Conosciamo il Transistor

La Figura 9-2 mostra lo schema elettrico ed il disegno pratico del transistor 2N3904. Ci sono molti tipi diversi di transistor. Questo è chiamato NPN, e si riferisce al tipo di materiali usato per fabbricare il transistor ed al modo in cui questi materiali sono stesi sul supporto. Il modo migliore per iniziare è di pensare al transistor come ad una valvola usata per controllare la corrente. Transistor differenti controllano l'ammontare della corrente che li attraversa in modi diversi. Questo transistor controlla quanta corrente entra dal Collettore (C) ed esce dall'Emettitore (E). Usa la quantità di corrente che scorre nella Base (B) per regolare la corrente che passa dal Collettore all'emettitore. Con una corrente molto piccola nella base B, una corrente di circa 416 volte più grande, scorre nel transistor entrando in C ed uscendo da E.

9

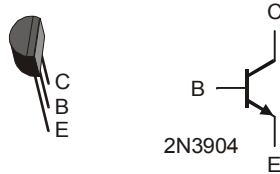



Figura 9-2
Transistor 2N3904

Il Datasheet del componente 2N3904: Come ricordato precedentemente, i produttori di semiconduttori pubblicano dei documenti chiamati datasheet (fogli dati) delle parti da loro costruite. Questi datasheet contengono informazioni che gli ingegneri usano per inserire il componente nei prodotti. Per vedere un esempio di datasheet di componente, per il 2N3904: Andate a www.fairchildsemi.com. Digitate 2N3904 nel campo di ricerca nella pagina iniziale della Fairchild Semiconductor, e clickate Go. Uno dei risultati della ricerca dovrebbe essere un link alla cartella del prodotto. Seguite il link della cartella del prodotto, quindi clickate Download questo link del Datasheet. Molti browser web visualizzano il datasheet aprendolo con Adobe Acrobat Reader.

Componenti per l'esempio del Transistor

(1) Transistor – 2N3904
(2) Resistenze – 100 kΩ (marrone-nero-giallo)
(1) LED – qualsiasi colore
(1) Potenziometro – 10 kΩ
(3) Ponticelli

Costruzione e prova del circuito del Transistor

La Figura 9-3 mostra un circuito che potete usare per controllare manualmente la quantità di corrente che il transistor fa scorrere nel LED. Ruotando la manopola del Potenziometro, il circuito invierà quantità diverse di corrente alla base del transistor. Questo causerà un cambiamento nella quantità di corrente che il transistor farà scorrere dal suo collettore al suo emettitore. Il LED vi darà una chiara indicazione del cambiamento accendendosi più o meno intensamente.

- √ Costruite il circuito mostrato in Figura 9-3.
- √ Ruotare la manopola del Potenziometro e verificate che il LED cambi di luminosità in risposta al cambiamento di posizione della spazzola del Potenziometro.

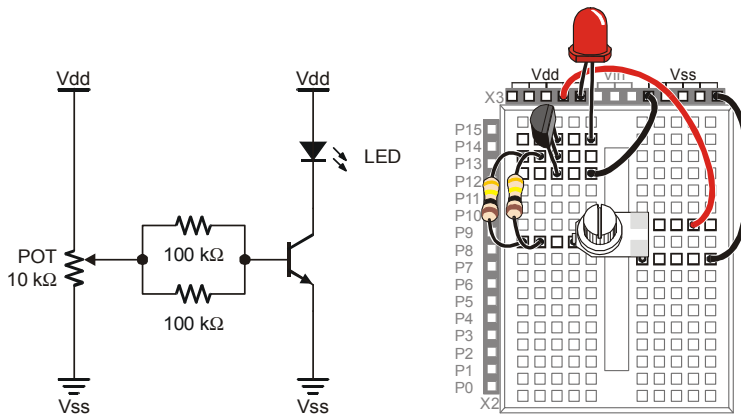



Figura 9-3
Circuito del transistor controllato manualmente con un potenziometro

Il Vostro Turno – Spengimento-Accensione del Transistor

Se tutto quello che volete fare è spengere ed accendere un transistor, potete usare il circuito mostrato in Figura 9-4. quando il BASIC Stamp invia un segnale alto a questo circuito, farà in modo di far scorrere tanta corrente quanta ne farebbe scorrere il potenziometro ruotato per la massima luminosità. Quando il BASIC Stamp invia un segnale basso a questo circuito, inibirà il transistor dal far scorrere corrente, ed il LED non emetterà luce.

Quale differenza c'è fra questo circuito ed il LED collegato direttamente ad un Pin I/O? I Pin I/O del BASIC Stamp hanno il limite di quanta corrente possono gestire. Anche i transistor hanno limitazioni, ma sono molto più alti. Nel testo *Industrial Control Student Guide*, è usato un transistor per pilotare una piccola ventola in DC. È anche usato per fornire molta corrente ad una piccola resistenza usata come elemento riscaldante. Ciascuna di queste due applicazioni assorbirebbero tanta corrente da danneggiare velocemente il BASIC Stamp, ma i transistor suppliscono alla carenza.

- √ Costruite il circuito mostrato in Figura 9-4.
- √ Scrivete un programma che invia due segnali alti e bassi al secondo a P8. SUGGERIMENTO: LedOnOff.bs2 dal Capitolo #2 deve essere modificato per inviare segnali alti /bassi a P8 invece che a P3. Ricordatevi di salvarlo con un nuovo nome, prima di fare le modifiche.
- √ Lanciate il programma e verificate che vi faccia accendere/spengere il LED.

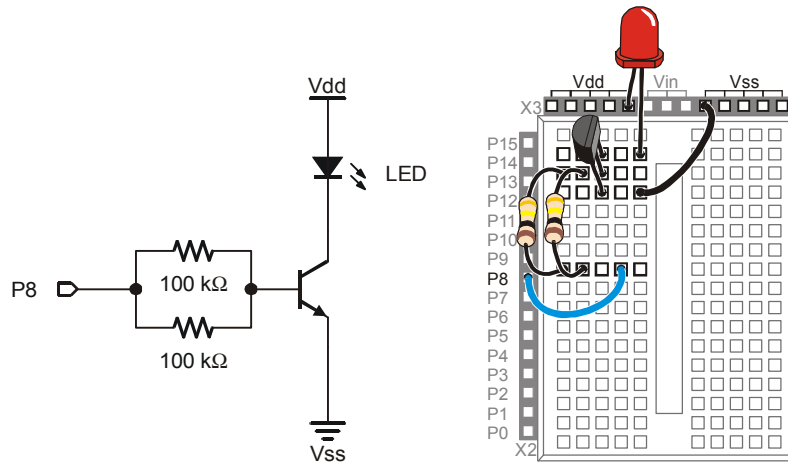



Figura 9-4
Circuito che da al Transistor del BASIC Stamp il Controllo On/Off

ESERCIZIO #2: PRESENTAZIONE DEL POTENZIOMETRO DIGITALE

In questo Esercizio, sostituirete il potenziometro a regolazione manuale con un potenziometro a circuito integrato che viene regolato digitalmente. Programmerete quindi il BASIC Stamp per regolare il potenziometro digitale, che a sua volta regolerà la luminosità del LED nello stesso modo del potenziometro manuale del precedente esercizio.

Conosciamo il Potenziometro Digitale

La Figura 9-5 mostra la piedinatura del Potenziometro digitale che userete in questo Esercizio. Questo chip ha 8 piedini, Quattro per ogni lato spaziati per facilitare l'inserimento nella piastra prototipi (distanziati di 1/10 di pollice). Il fabbricante mette un punto di riferimento sul contenitore plastico in modo che possiate distinguere la differenza tra il piedino 1 ed il piedino 5. Il punto di riferimento è un piccolo semicerchio sul contenitore del chip. Potete usarlo per trovare il numero di piedino del chip, il numero di piedino, si conta dall'alto in senso antiorario a partire dal punto di riferimento.

Sostituzione dei componenti: La Parallax è qualche volta costretta a cambiare il componente. Esso funzionerà nello stesso modo, ma l'etichetta può essere diversa. Se nel kit dei componenti di Che cosa è un Microcontrollore trovate che il potenziometro digitale non è etichettato AD5220, state comunque tranquilli che funzionerà nello stesso modo e correttamente in questo Esercizio.

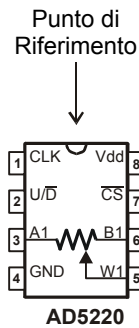



Figura 9-5
Piedinatura del AD5220

Usate il punto di riferimento per disporre l'AD5220 come mostrato nel disegno quando lo inserite nel vostro circuito o nella piastra prototipi.

Sommario dei piedini e delle relative funzioni dell'AD5220:

1. CLK – Il piedino che riceve gli impulsi di temporizzazione (segnali alto-Basso-Alto) per il movimento della spazzola del potenziometro.
2. U/D – Il piedino che riceve un segnale alto per muovere la spazzola (W1) verso il terminale A1, ed un segnale basso per muovere la spazzola verso B1. questo piedino imposta la direzione, verso cui il terminale si muoverà al ricevimento dell'impulso (segnale Alto-Basso-Alto) sul piedino CLK.
3. A1 – Il terminale A del Potenziometro.
4. GND – La connessione di massa. La massa sulla Board of Education e sulla BASIC Stamp HomeWork Board è il terminale Vss.
5. W1 – Il terminale (W) della spazzola del Potenziometro.
6. B1 – Il terminale B del Potenziometro.
7. CS – Il piedino di selezione del chip. Applicate un segnale alto a questo piedino, ed il chip ignorerà qualsiasi comando inviato ai piedini CLK ed U/D. Applicate un segnale basso a questo piedino, ed il chip sarà pienamente operativo.
8. Vdd – Collegato a +5 V, che sulla Board of Education e sulla BASIC Stamp HomeWork Board è Vdd.

9

Il Datasheet del componente AD5220: Per vedere il datasheet dell'AD5220: Andate al sito www.analog.com. Digitate AD5220 nel campo di ricerca della pagina iniziale della Analog Devices, e clickate il pulsante di ricerca. Clickate il link dei DataSheet. Clickate il link dell'AD5220: Increment/Decrement Digital Potenziometro Datasheet".

Componenti del circuito a transistor controllato dal potenziometro Digitale

(1) Transistor – 2N3904
(2) Resistenze – 100 kΩ (marrone-nero-giallo)
(1) LED – qualsiasi colore
(1) Potenziometro Digitale– AD5220

Costruzione del circuito del Potenziometro Digitale

La Figura 9-6 mostra lo schema elettrico del circuito del potenziometro digitale usato al posto del Potenziometro manuale, e la Figura 9-7 mostra lo schema di cablaggio del circuito. Il BASIC Stamp può controllare il Potenziometro digitale emettendo segnali di controllo da P5 e da P6.

√ Costruite il circuito mostrato in Figura 9-6 ed in Figura 9-7.

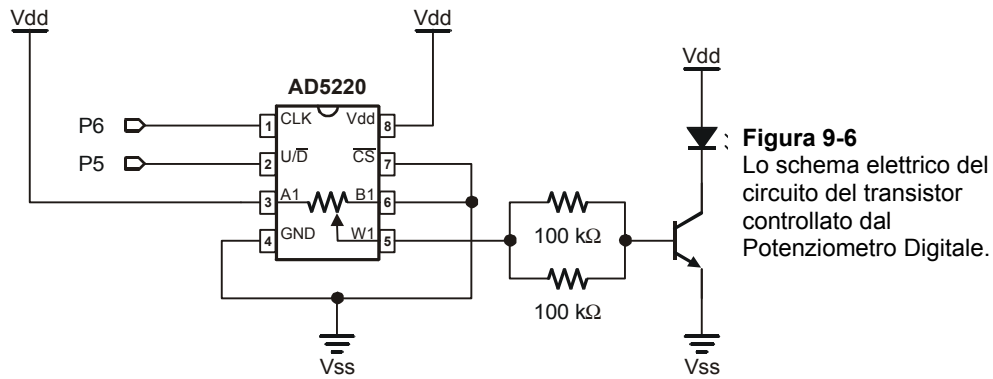



Figura 9-6
Lo schema elettrico del circuito del transistor controllato dal Potenziometro Digitale.

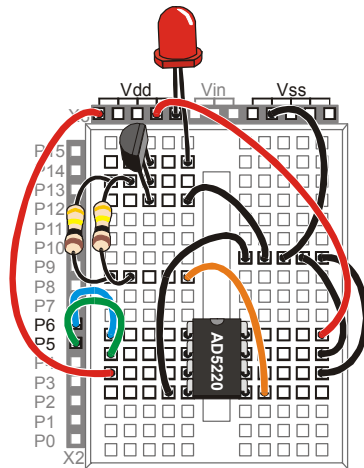



Figura 9-7
Schema di cablaggio del transistor controllato dal Potenziometro Digitale.

Programmazione del controllo del Potenziometro Digitale

Immaginate che la manopola del Potenziometro manuale dell'esercizio precedente abbia 128. Immaginate anche che il Potenziometro sia nella sua posizione intermedia. Questo significa che potete ruotare la manopola per 63 passi in una direzione e per 64 passi nell'altra direzione.

Assumete di ruotare la manopola del Potenziometro di un passo in senso orario. Il LED si aumenterà debolmente la sua luminosità. Questo è lo stesso che inviare un segnale alto al pin U/D dell'AD5220 ed un impulso (Alto-Basso-Alto) al pin CLK.

```
HIGH 5
PULSOUT 6, 1
```

Immaginate quindi che ruotiate il vostro Potenziometro manuale di 3 passi in senso antiorario. La luminosità del LED si affievolirà un poco. Questo sarebbe lo stesso che inviare un segnale basso al piedino U/D dell'AD5220 e quindi inviare tre impulsi al pin CLK.

```
LOW 5
FOR counter = 1 TO 3
  PULSOUT 6, 1
  PAUSE 1
NEXT
```

Immaginate poi che ruotiate il Potenziometro tutto in senso orario. È lo stesso che inviare un segnale alto al pin U/D dell'AD5220 ed inviare 65 impulsi al pin CLK. Ora il LED dovrebbe essere alla massima luminosità.

```
HIGH 5
FOR counter = 1 TO 65
  PULSOUT 6, 1
  PAUSE 1
NEXT
```


Infine, immaginate di ruotare il vostro Potenziometro manuale tutto in senso antiorario. Il LED dovrebbe essere spento. È lo stesso che inviare un segnale basso al pin U/D ed applicare 128 impulsi al pin CLK.

```
LOW 5
FOR counter = 0 TO 127
  PULSOUT 6, 1
  PAUSE 1
NEXT
```

Programma di Esempio: DigitalPotUpDown.bs2

Questo programma esempio regola il Potenziometro in un senso e nell'altro, da un estremo all'altro, facendo gradualmente illuminare ed attenuare la luminosità del LED.

√ Digitate e lanciateDigitalPotUpDown.bs2.

```
' Che cosa è un Microcontrollore- DigitalPotUpDown.bs2
' Fa scorrere il potenziometro digitale fra i suoi estremi.

' {$STAMP BS2}
' {$PBASIC 2.5}
```

```
counter        VAR     Byte

DO

  LOW 5

  FOR counter = 0 TO 127
    PULSOUT 6, 1
    PAUSE 10
  NEXT

  HIGH 5

  FOR counter = 0 TO 127
    PULSOUT 6, 1
    PAUSE 10
  NEXT

LOOP
```

Il Vostro Turno – Cambiare la Velocità e Condensare il Codice.

Potete aumentare o diminuire la velocità con cui la luminosità del LED aumenta e diminuisce, variando l'argomento ***`Duration`*** del comando **`PAUSE`**.

√ Modificate e rilanciate il programma usando **`PAUSE 20`** ed annotate la differente velocità di variazione di luminosità del LED.
√ Ripetete con **`PAUSE 5`**.

Per semplificare questo programma potete anche usare il comando **`TOGGLE`**. Esso cambia lo stato di un pin I/O del BASIC Stamp. Se il pin I/O stava inviando un segnale alto, **`TOGGLE`** gli fa inviare un segnale basso. Se il pin I/O stava inviando un segnale basso, **`TOGGLE`** gli fa inviare un segnale alto.

√ Salvate DigitalPotUpDown.bs2 come DigitalPotUpDownWithToggle.bs2.
√ Modificate il programma in modo che sia simile a quello sotto elencato.
√ Lanciate il programma e verificate che funzioni nello stesso modo del programma DigitalPotUpDown.bs2.
√ Confrontate il numero delle linee di codice che occorrono per fare lo stesso lavoro.

Rimanere a corto di memoria programma è un problema in cui molte persone si imbattono quando I loro progetti con il BASIC Stamp diventano grandi e complicati. Usando **TOGGLE** invece di due cicli **FOR…NEXT** è solamente un esempio delle molte tecniche che possono essere usate per fare lo stesso lavoro con la metà del codice.

```
' Che cosa è un Microcontrollore- DigitalPotUpDownWithToggle.bs2
' Fa scorrere il potenziometro digitale fra i suoi estremi.

' {$STAMP BS2}
' {$PBASIC 2.5}

counter        VAR     Byte

LOW 5

DO
  FOR counter = 0 TO 127
    PULSOUT 6,5
    PAUSE 10
  NEXT

  TOGGLE 5
LOOP
```

9

Uno Sguardo all'interno del Potenziometro Digitale

La Figura 9-8 mostra lo schema elettrico del Potenziometro interno all'AD5220. L'AD5220 ha 128 elementi resistivi, ciascuno dei quali è 78.125 Ω (valore nominale). La somma di tutti i 128 elementi è di 10,000 Ω o 10 kΩ.

Il valore nominale: I componenti come le Resistenze ed i condensatori normalmente hanno un valore nominale ed una tolleranza. Ciascuno degli elementi resistivi dell'AD5220 ha un valore nominale di 78.125 Ω, con una tolleranza del 30% (23.438 Ω) al di sopra od al di sotto del valore nominale.

Tra ciascuno di questi elementi resistivi c'è un'interruttore. Ogni interruttore in realtà è un gruppo di transistor che vengono accesi o spenti per far passare o meno la corrente. Solamente un interruttore alla volta può essere chiuso. Se uno degli interruttori superiori è chiuso (come le posizioni 125, 126, o 127), è come avere la manopola del Potenziometro manuale ruotata quasi tutta in senso orario. Se è chiusa la posizione 0 o 1, è come avere il Potenziometro manuale quasi tutto ruotato in senso antiorario.

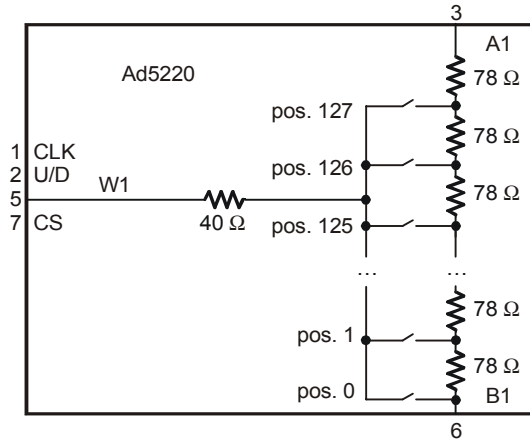



Figura 9-8
All'interno dell'AD5220

Immaginate che sia chiusa la Pos. 126. Se la volete impostare a 125, (aprire la pos. 126 e chiudere la pos. 125), impostate basso U/D, quindi applicate un impulso a CLK. Se volete impostare la pos. 127, mettete alto U/D, ed applicate 2 impulsi. Se lo volete portarlo ad 1, mettete U/D basso, ed applicate 126 impulsi.

Il prossimo programma esempio usa il Terminale di Debug per chiedervi quale impostazione desiderate. Quindi decide se impostare U/D alto o basso, ed applica il giusto numero di impulse per muoversi dalla vecchia alla nuova impostazione.

Ad eccezione delle EEPROM Data, il prossimo programma esempio ha tutte le sezioni che vi aspettereste di trovare in un programma applicativo:

- Titolo – commenti che includono il nome del file di un programma, la sua descrizione, e le direttive Stamp e PBASIC.
- EEPROM Data – La dichiarazione **DATA** usata dal programma.
- Definitioni I/O– costanti, dichiarazioni che definiscono il numero dei pin I/O.
- Costanti – dichiarazioni delle che definiscono altri valori usati nel programma.
- Variabili – Dichiarazione delle variabili.
- Inizializzazione – Una routine chef a partire il programma nel modo giusto. Nel prossimo programma, il cursore del Potenziometro deve essere portato a zero.
- Principale – La routine che gestisce i compiti primari che il programma deve svolgere.
- Subroutine – I segmenti di codice che svolgono compiti specifici, sia per altre subroutine che per la routine principale.

Programma di Esempio: TerminalControlledDigtialPot.bs2

Potete usare questo programma esempio ed il Terminale di Debug per impostare la posizione del potenziometro digitale. Cambiando l'impostazione del potenziometro digitale, potrete cambiare la luminosità del LED collegato al transistor controllato dal BASIC Stamp. La Figura 9-9 mostra un esempio di inserimento del valore 120 nella finestra di trasmissione del Terminale di Debug mentre il programma sta girando. Dal momento che la vecchia impostazione era 65, quando verrà impostato a 120, il LED si illuminerà a quasi il doppio della luminosità precedente.

Finestre

Trasmissione →

Ricezione →

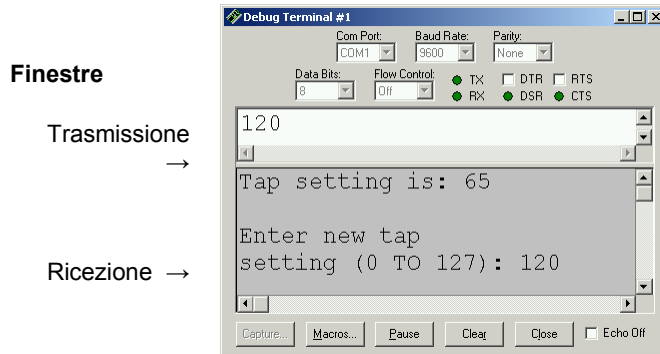



Figura 9-9
Invio di messaggi al BASIC Stamp

Clickate nella finestra di trasmissione (superiore) ed inserite i numeri delle nuove impostazioni.

9

√ Digitate e lanciateTerminalControlledDigtialPot.bs2.

√ Digitate valori tra 0 e 127 nel Terminale di Debug. Assicuratevi di premere Enter dopo aver digitato le cifre.

```
' -----[ Title ]-----------------------------------------------------------
' Che cosa è un Microcontrollore- TerminalControlledDigitalPot.bs2
' Aggiorna la posizione del potenziometro digitale con il Terminale di Debug.

' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ EEPROM Data ]-----------------------------------------------------


' -----[ I/O Definitions ]-------------------------------------------------

UdPin          CON     5                      ' Imposta I valori dei pin I/O
ClkPin         CON     6                      ' collegati a CLK e ad U/D.

' -----[ Constants ]-------------------------------------------------------
```

```
DelayPulses    CON     10       ' Ritardo per osservare la variazione del LED.
DelayReader    CON     2000

' -----[ Variables ]----------------------------------------------------

counter        VAR     Byte     ' Contatore del ciclo FOR...NEXT.
oldTapSetting  VAR     Byte     ' Impostazione precedente.
newTapSetting  VAR     Byte     ' Nuova impostazione.

' -----[ Initialization ]-----------------------------------------------

oldTapSetting = 0               ' Inizializza la nuova e la vecchia
newTapSetting = 0               ' impostazione a zero.

LOW UdPin                       ' Imposta il pin U/D basso.
FOR counter = 0 TO 127          ' Potenziometro al suo livello più basso.
  PULSOUT 6,5
  PAUSE 1
NEXT

' -----[ Main Routine ]-------------------------------------------------

DO:

  GOSUB Get_New_Tap_Setting     ' visualizza utente e prende impostazione.
  GOSUB Set_Ud_Pin              ' imposta pin U/D per sinistra/destra.
  GOSUB Pulse_Clk_pin           ' Invia gli impulsi.

LOOP

' -----[ Subroutines ]--------------------------------------------------

Get_New_Tap_Setting:                   ' visualizza le Istruzioni e
                                       ' prende l'impostazione dell'utente
  DEBUG CLS, "Tap setting is: ",       ' per il valore del potenziometro.
        DEC newTapSetting, CR, CR
  DEBUG "Enter new tap", CR, "setting (0 TO 127): "
  DEBUGIN DEC newTapSetting

  RETURN

Set_Ud_Pin:                            ' Esamina il nuovo ed il vecchio valore
                                         ' per decidere il valore del pin U/D.
  IF newTapSetting > oldTapSetting THEN  ' Avvisa l'utente se i valori sono
    HIGH UdPin                           ' uguali.
    oldTapSetting = oldTapSetting + 1    ' Incrementa Pulse_Clk_pin.
  ELSEIF newTapSetting < oldTapSetting THEN
    LOW UdPin
    oldTapSetting = oldTapSetting - 1    ' Decrementa Pulse_Clk_pin.

  ELSE
```

```
    DEBUG CR, "New and old settings", CR,
              "are the same, try ", CR,
              "again...", CR
    PAUSE DelayReader                        ' Da al lettore il tempo per
  ENDIF                                      ' vedere il Messagio.

  RETURN

Pulse_Clk_pin:

  ' Invia gli impulsi per andare dal vecchio al nuovo valore.
  ' Ricordate che Set_Ud_Pin varia dal valore di oldTapSetting verso
  ' newTapSetting di uno. Questo evita che il ciclo FOR...NEXT sia eseguito
  ' una volta di troppo.

  FOR counter = oldTapSetting TO newTapSetting
    PULSOUT ClkPin, 1
    PAUSE DelayPulses
  NEXT

  oldTapSetting = newTapSetting              ' Tiene traccia del vecchio e del
                                             ' nuovo valore di impostazione.
  RETURN
```

9

SOMMARIO

Questo Capitolo ha presentato i circuiti integrati e mostrato un esempio di come possono essere usati con il BASIC Stamp. È stato usato un transistor come regolatore di corrente, ed è stato usato un potenziometro digitale per controllare la quantità di corrente da far passare nel transistor.

Domande

1. Quali sono i nomi dei terminali del transistor che avete usato in questo capitolo?
2. Quale terminale controlla la corrente che passa attraverso il transistor?
3. Che cosa potete fare per aumentare o diminuire la corrente passante attraverso il transistor?
4. Che cos'è la piedinatura?
5. Che cosa è il punto di riferimento?

Esercizi

1. Scrivete un segmento di codice che faccia inviare da un pin I/O BASIC Stamp segnali alti/bassi dieci volte al secondo usando il comando **TOGGLE**.
2. Scrivete un segmento di codice che regoli il cursore del potenziometro digitale nella posizione 0 senza curarsi di quale è la sua posizione iniziale.

3. Scrivete un segmento di codice che imposti il cursore del potenziometro digitale alla posizione 32, quindi alla posizione 64, poi alla posizione 96, ed infine alla posizione 127. Il potenziometro digitale dovrebbe rimanere in ciascuna impostazione per un secondo.

Progetti

1. Aggiungete al vostro progetto una fotoresistenza in modo che regoli la luminosità di un LED in base alla luminosità vista dalla stessa.
2. Collegare il pin CS dell'AD5220 ad un pin I/O del BASIC Stamp ed usatelo per controllare la possibilità o meno di fargli accettare segnali di controllo dal BASIC Stamp. Dovrete scollegare il pin CS da Vss prima di collegarlo al pin I/O del BASIC Stamp. Ricordate che un segnale basso attiva il chip ed un segnale alto lo disattiva.

Ulteriori Approfondimenti

"*Industrial Control*", Student Guide, Version 2.0, Parallax Inc., 2002

Industrial Control usa il transistor come interruttore on/off per un elemento riscaldante resistivo. Usa anche un fototransistor per rivelare il passaggio dei raggi di una ruota.

Capitolo #10: Far girare tutto lo Spettacolo

INTEGRAZIONE DEI SOTTOSISTEMI

La maggior parte degli esercizi in questo testo insegna come programmare il BASIC Stamp per interagire con uno o due circuiti alla volta. Molti microcontrollori devono gestire decine od anche centinaia di circuiti. Questo Capitolo vi spiegherà alcune delle tecniche usate per gestire una varietà di circuiti con un singolo microcontrollore. La programmazione dei microcontrollori per orchestrare le attività di circuiti differenti che eseguono funzioni uniche, è chiamata integrazione di sottosistema.

L'esempio di questo capitolo usa un assieme di sensori consistente in due circuiti tasto, un circuito potenziometrico ed un circuito a fotoresistenza. Gli esercizi di questo capitolo vi guideranno nella costruzione e nel collaudo individuale di ciascun sottosistema. Dopo aver costruito e collaudato ciascun sottosistema, scriverete un programma che riunisce ciascun sottosistema in un sistema completo. La Figura 10-1 mostra lo schema elettrico del sistema che costruirete. L'elenco principale dei componenti è mostrato sotto.

Ricordate sempre: quando possibile collaudate individualmente ciascun sottosistema prima di provare a farli lavorare insieme. Se seguirete questa regola, i vostri progetti funzioneranno in modo più appropriato e verranno aumentate grandemente le possibilità di successo. Gli esercizi di questo capitolo vi guideranno attraverso il procedimento.

Elenco dei Componenti dell'assieme di Sensori

(4) Resistenze – 220 Ω (rosso-rosso-marrone)
(2) Resistenze – 10 kΩ (marrone-nero-arancio)
(2) Tasti – normalmente aperti
(2) Condensatori – 0.1 μF
(1) Potenziometro – 10 kΩ
(1) Fotoresistenza
(5) Ponticelli

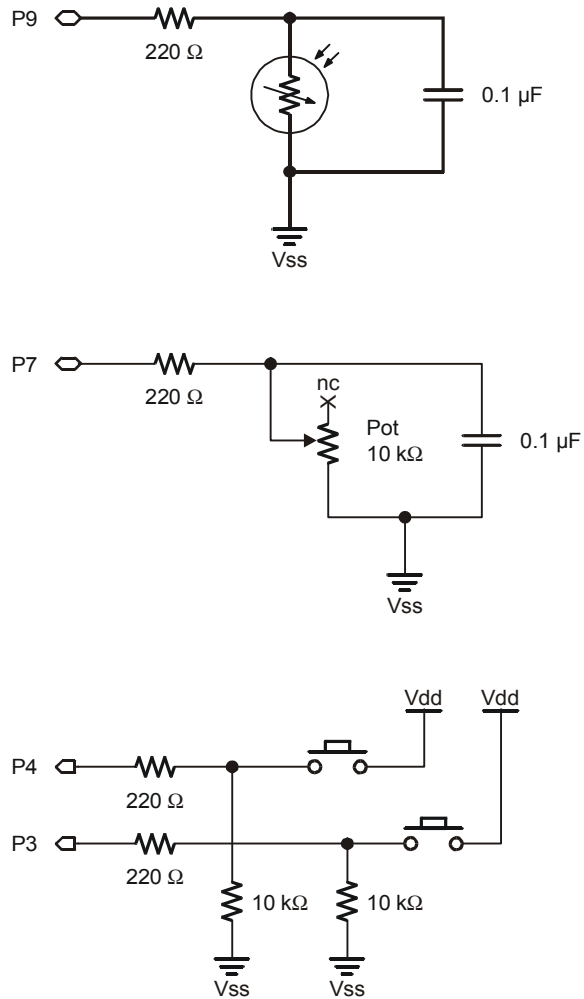



Figura 10-1
Schema Elettrico dell'assieme di Sensori

ESERCIZIO #1: COSTRUZIONE E COLLAUDO DI CIASCUN CIRCUITO TASTO

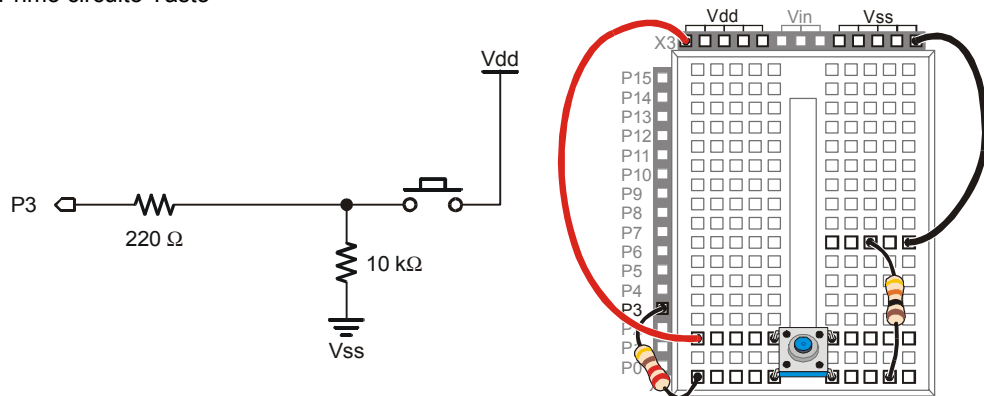
Questo Esercizio inizia con la costruzione ed il collaudo dei singoli circuiti tasto. Appena siete sicuri che il primo circuito funziona correttamente, potrete cominciare a costruire e collaudare il secondo circuito tasto.

Componenti dei circuiti Tasto

(2) Tasti – normalmente aperti
(2) Resistenze – 10 kΩ (marrone-nero-arancio)
(2) Resistenze – 220 Ω (rosso-rosso-marrone)
(2) Ponticelli

Costruzione del primo circuito Tasto

Costruite il circuito tasto mostrato in Figura 10-2.

Figura 10-2
Primo circuito Tasto

Collaudo del primo circuito Tasto

Scrivere un programma per il collaudo di questo circuito dovrebbe essere molto semplice, specialmente ora che lo avete già fatto in precedenza.

Programma di Esempio: ReadTastoState.bs2

Questa è una ripetizione del programma di collaudo dei tasti del Capitolo #3:

- √ Digitate e lanciate ReadTastoState.bs2.
- √ Verificate che il BASIC Stamp sia in grado di leggere lo stato del tasto.
- √ Prima di andare al prossimo circuito correggete ogni problema che incontrate.

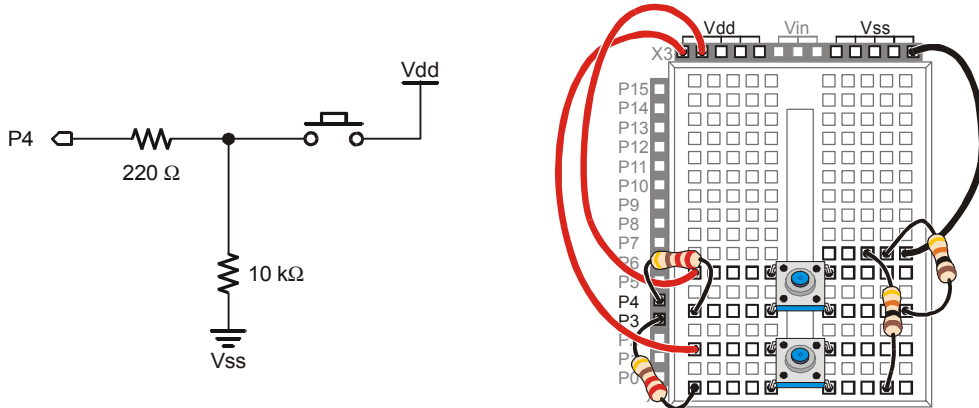
```
' Che cosa è un Microcontrollore- ReadTastoState.bs2
' Controlla ed invia lo stato del tasto al Terminale di Debug ogni 1/4 di
```

```
 ' secondo.

' {$STAMP BS2}
' {$PBASIC 2.5}

DO

  DEBUG ? IN3
  PAUSE 250

LOOP
```

Il Vostro Turno – Costruzione e Collaudo del Secondo circuito Tasto

Una volta che il primo circuito tasto è stato costruito e collaudato, il procedimento può essere ripetuto per il secondo circuito tasto.

- √ Aggiungete il circuito tasto mostrato in Figura 10-3.
- √ Collaudate il secondo circuito tasto modificando ReadTastoState.bs2 in modo che legga il circuito collegato a P4. Lanciate il programma e verificate che il secondo tasto funzioni.
- √ Prima di andare al prossimo circuito correggete ogni problema che incontrate.

Figura 10-3
Aggiunta del Secondo Circuito Tasto al Progetto

ESERCIZIO #2: COSTRUZIONE E COLLAUDO DI CIASCUN CIRCUITO RC-TIME

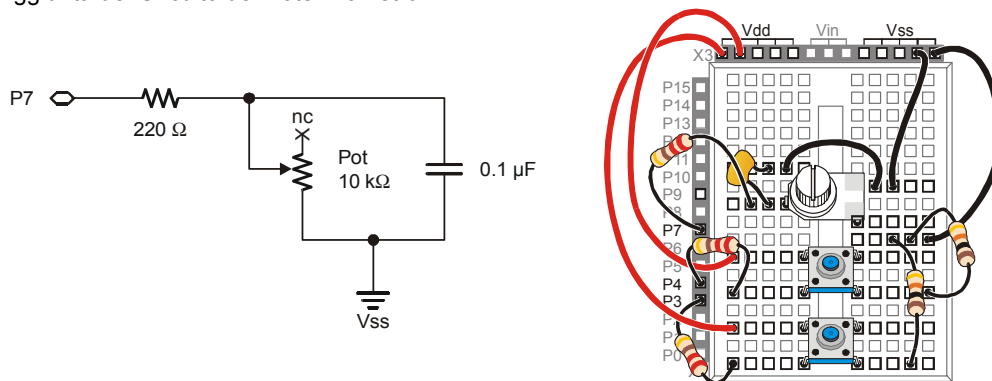
Ora che i due circuiti tasto sono stati montati e collaudati, potete passare ai circuiti RC-time.

Componenti Aggiuntivi per i Circuiti della Fotoresistenza e del Potenziometro

(2) Resistenze – 220 Ω (rosso-rosso-marrone)
(2) Condensatori – 0.1 µF
(4) Ponticelli
(1) Fotoresistenza
(1) Potenziometro – 10 kΩ

Montaggio del Circuito del Potenziometro

√ Aggiungere al progetto il circuito del Potenziometro mostrato in Figura 10-4.

Figura 10-4
Aggiunta del Circuito del Potenziometro

10

Collaudo del Circuito del Potenziometro

Potete collaudare questo circuito usando ReadPotWithRcTime.bs2. Questo programma è stato usato per la prima volta nel Chapter #5 per leggere il circuito del Potenziometro.

Programma di Esempio: ReadPotWithRcTime.bs2

√ Digitate e lanciate ReadPotWithRcTime.bs2.

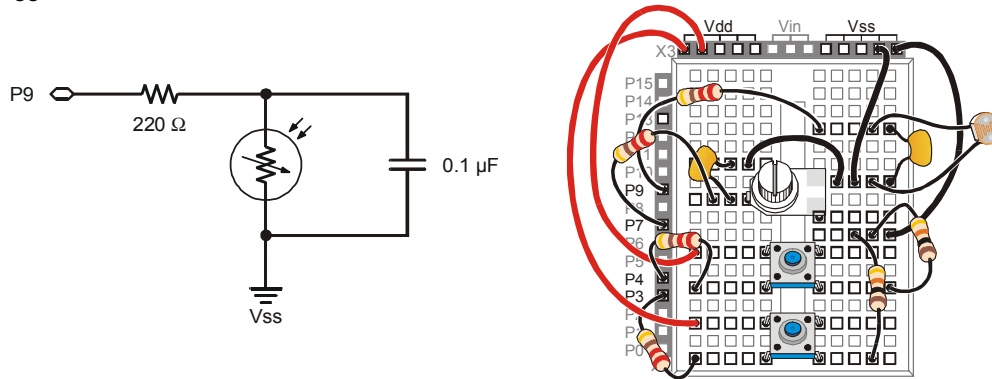
√ Verificate che il BASIC Stamp ottenga misure attendibili dal Potenziometro.
√ Prima di andare al prossimo circuito correggete ogni problema che incontrate.

```
' Che cosa è un Microcontrollore- ReadPotWithRcTime.bs2
' Legge il Potenziometro in un circuito RC-time usando il comando RCTIME.

' {$STAMP BS2}
' {$PBASIC 2.5}

time          VAR     Word

DO

  HIGH 7
  PAUSE 100
  RCTIME 7, 1, time
  DEBUG HOME, " time =  ", DEC5 time

LOOP
```

Il Vostro Turno – Costruzione e collaudo del Circuito della Fotoresistenza

Una volta che il primo circuito RC-time è stato montato e collaudato, il procedimento può essere ripetuto per il secondo circuito RC-time.

√ Aggiungete sulla scheda prototipi il circuito della fotoresistenza mostrato in Figura 10-5 al vostro progetto.
√ Modificate ReadPotWithRcTime.bs2 collegando la fotoresistenza a P9.
√ Prima di andare al prossimo circuito correggete ogni problema che incontrate.

Figura 10-5
Aggiunta del Circuito della Fotoresistenza

ESERCIZIO #3: ESEMPIO DI INTEGRAZIONE DEL SOTTOSISTEMA

Ora che tutti gli elementi dell'assieme di sensori sono stati montati e collaudati, potete scrivere un programma che usa tutti e quattro i circuiti dei sensori. Questo esempio dimostrerà come programmare il BASIC Stamp per visualizzare un terminale che un tecnico potrà usare per tenere sotto controllo i sensori.

Programmazione di un Sistema di Menù che fa Uso della Direttiva di Input/Output dei PIN

Il comando **ON…GOSUB** può essere molto utile per i menu.

ON *offset* , GOSUB *Target1, {Target2, Target3,…}*

Nel prossimo programma esempio, il comando **ON...GOSUB** usa il valore di una variabile chiamata **request** per indirizzarlo ad una di quattro subroutine. Se il valore di **request** è 0, il programma esegue un **GOSUB Read_Tasto_1**. Se il valore di **request** è 1, il programma esegue un **GOSUB Read_Tasto_2**, e così via.

```
ON request GOSUB Read_Tasto_1, Read_Tasto_2,
                 Read_Pot, Read_Fotoresistenza
```

Potete usare la direttiva **PIN** per dare un nome a ciascun pin I/O che userete in un programma PBASIC. L'Editor del BASIC Stamp deciderà quindi se userete quei pin I/O come ingressi o come uscite (o tutte e due). La sintassi della direttiva **PIN** è:

PinName* PIN *PinNumber

Il prossimo programma esempio mostra come potete dichiarare e quindi usare un ***PinName***. Per esempio, l'Editor del BASIC Stamp presume che vogliate visualizzare il valore di ingresso di un pin I/O (**IN3**) se state usando in un comando **DEBUG, Pb1Pin** ***PinName***. Questo comando **DEBUG** visualizzerà un 1 o uno 0 a secondo se il tasto collegato a P3 è premuto oppure No. Perché? Perché l'Editor del BASIC Stamp sa che deve sostituire **Pb1Pin** con **IN3**.

```
Pb1Pin         PIN     3
DEBUG CLS, ? Pb1Pin
```

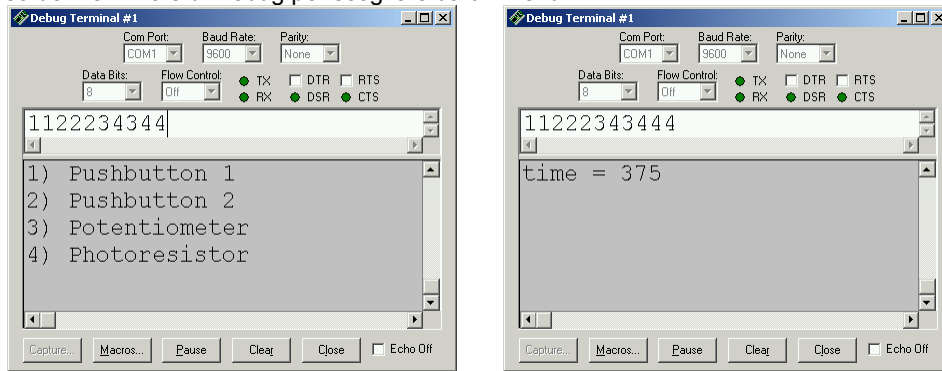
Qui c'è un altro esempio dove l'Editor del BASIC Stamp sa che deve usare il valore **9** perché **PhotoPin** ***PinName*** è usato in un comando **HIGH** ed in un comando **RCTIME**.

```
PhotoPin       PIN     9
HIGH PhotoPin
RCTIME PhotoPin, 1, time
```

Programma di Esempio: TerminalOperatedSensorArray.bs2

La Figura 10-6 mostra il Terminale di Debug visualizzato dal programma TerminalOperatedSensorArray.bs2. Sulla sinistra è mostrato il menu principale, e sulla destra un esempio di visualizzazione quando è selezionato '4'. Ricordatevi di usare la finestra superiore per inviare le vostre selezioni al BASIC Stamp.

√ Digitate e lanciateTerminalOperatedSensorArray.bs2.
√ Clickate nella finestra superiore del Terminale di Debug e digitate le cifre per fare la vostra scelta nel menu.
√ La misura visualizzata è quella presa nell'istante in cui è stata fatta la scelta nel menù, ed è visualizzata per un secondo e mezzo. Ricordatevelo quando premete e tenete premuti i tasti, regolate il potenziometro ed ombreggiate la fotoresistenza.

Figura 10-6
Uso del Terminale di Debug per scegliere da un menu

```
' -----[ Title ]-----------------------------------------------------------
' Che cosa è un Microcontrollore- TerminalOperatedSensorArray.bs2
' Usa il Terminale di Debug per scegliere di leggere uno dei quattro sensori.

' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ I/O Definitions ]-------------------------------------------------

Pb1Pin         PIN     3
Pb2Pin         PIN     4
PotPin         PIN     7
PhotoPin       PIN     9

' -----[ Constants ]-------------------------------------------------------

DelayRc        CON     100
DelayReader    CON     1500

' -----[ Variables ]-------------------------------------------------------

request        VAR     Nib
time           VAR     Word

' -----[ Main Routine ]----------------------------------------------------

DO:

  GOSUB Display_Menu
  GOSUB Get_Request
  ON request GOSUB Read_Tasto_1, Read_Tasto_2,
                   Read_Pot, Read_Fotoresistenza
```

```
LOOP

' -----[ Subroutine - Display_Menu ]---------------------------------------

Display_Menu:

  DEBUG CLS, "MENU: ", CR, CR,
             "1) Tasto 1", CR,
             "2) Tasto 2", CR,
             "3) Potenziometro", CR,
             "4) Fotoresistenza", CR, CR
  RETURN

' -----[ Subroutine - Get_Request ]----------------------------------------

Get_Request:

  DEBUGIN DEC1 request
  request = request - 1

  RETURN

' -----[ Subroutine - Read_Tasto_1 ]----------------------------------

Read_Tasto_1:

  DEBUG CLS, ? Pb1Pin
  PAUSE DelayReader

  RETURN


' -----[ Subroutine - Read_Tasto_2 ]----------------------------------

Read_Tasto_2:

  DEBUG CLS, ? Pb2Pin
  PAUSE DelayReader

  RETURN

' -----[ Subroutine - Read_Pot ]-------------------------------------------

Read_Pot:

  HIGH PotPin
  PAUSE DelayRc
  RCTIME PotPin, 1, time
  DEBUG CLS, ? time, "     "
  PAUSE DelayReader
```

```
  RETURN

' -----[ Subroutine - Read_Fotoresistenza ]--------------------------------

Read_Fotoresistenza:

  HIGH PhotoPin
  PAUSE DelayRc
  RCTIME PhotoPin, 1, time
  DEBUG CLS, ? time, "     "
  PAUSE DelayReader

  RETURN
```

Il Vostro Turno – Modificare le Subroutine

I Terminali non sono le sole applicazioni del comando **ON…GOSUB**. Per esempio, se volete chiamare le quattro diverse subroutine una dopo l'altra, qui c'è un modo per farlo:

```
DO:

  ' GOSUB Display_Menu
  ' GOSUB Get_Request
  FOR request = 0 TO 3
    ON request GOSUB Read_Tasto_1, Read_Tasto_2,
                     Read_Pot, Read_Fotoresistenza
  NEXT

LOOP
```

Notare che le due subroutine che eseguono il compito di creare il terminale sono apostrofate ed è stato inserito in un ciclo **FOR…NEXT** il comando **ON…GOSUB**.

- √ Salvate il vostro programma esempio con il nuovo nome di: TerminalOperatedSensorArrayYourTurn.bs2.
- √ Modificate la routine principale facendo i cambiamenti appena discussi.
- √ Lanciate il programma e verificate che effettui tutte le misure.

ESERCIZIO #4: SVILUPPO ED AGGIUNTA DI SOTTOSISTEMI SOFTWARE

Diciamo che il vostro assieme di sensori è messo in un dispositivo che solamente alcuni dei vostri dipendenti sono abilitati ad usare, e per farlo devono fornire una password. Questo significa che dovrete espandere il vostro programma in modo che salvi e verifichi

la password. È meglio prima scrivere il programma password in modo che funzioni da solo, e poi aggiungerlo al vostro programma principale.

Programmazione di un controllore di Password

Potete salvare la password in un programma PBASIC usando la direttiva **DATA**. Per esempio:

```
Password       DATA    "pass!"
```

Lo stesso di: Questa direttiva **DATA** ha lo stesso significato che **Password DATA "p", "a", "s", "s", "!"**.

Per memorizzare altri valori nel programma, vi serviranno alcune variabili:

```
index          VAR     Nib
temp           VAR     Byte
```

Se state usando una password di cinque caratteri, c'è un genere particolare di dichiarazione (chiamata dichiarazione di array) che potete usare per dichiarare cinque variabili con lo stesso nome.

```
userEntry      VAR     Byte(5)
```

A questo punto avete una variabile a cinque byte chiamata userEntry: **userEntry(0)**, **userEntry(1)**, **userEntry(2)**, **userEntry(3)**, ed **userEntry(4)**.

Il comando **DEBUGIN** ha un formattatore chiamato **STR** che carica automaticamente I caratteri in un array. Per esempio, potete usare:

```
DEBUGIN STR userEntry \5
```

Se digitate cinque caratteri nella finestra di trasmissione del Terminale di Debug, il primo verrà messo in **userEntry(0)**, il secondo verrà messo in **userEntry(1)**, etc.

Esiste una parola chiave in PBASIC chiamata **EXIT** che potete usare per interrompere un ciclo. Per controllare una password, potete usare un istruzione **IF…THEN** con un comando **EXIT** per forzare l'interruzione prematura del ciclo se non tutti i caratteri sono identici. Quando il ciclo termina anzitempo, significa che **index** non ha contato fino a cinque, il che a sua volta significa che la password non era corretta:

```
FOR index = 0 TO 4
  READ Password + index, temp
  IF temp <> userEntry(index) THEN EXIT
```

```
NEXT

IF index <> 5 THEN
  DEBUG CR,"Password not correct.", CR
ENDIF
```

Il prossimo programma esempio mette il comando **`DEBUGIN, il ciclo FOR…NEXT`**, e l'istruzione **`IF…THEN`** all'interno del ciclo **`DO…LOOP UNTIL`** che continua l'esecuzione fino a che il valore di index arriva a 5 (indicando che è stata digitata la password corretta).

Programma di Esempio: PasswordChecker.bs2

La Figura 10-7 mostra il Terminale di Debug visualizzato da PasswordChecker.bs2. Quando lanciato, questo programma attende che digitiate le lettere "pass!" in risposta alla richiesta di password.

- √ Digitate e lanciatePasswordChecker.bs2.
- √ Provate a digitare alcune combinazioni di lettere che non siano la password, digitate quindi la combinazione di lettere che compongono la password: "pass!".

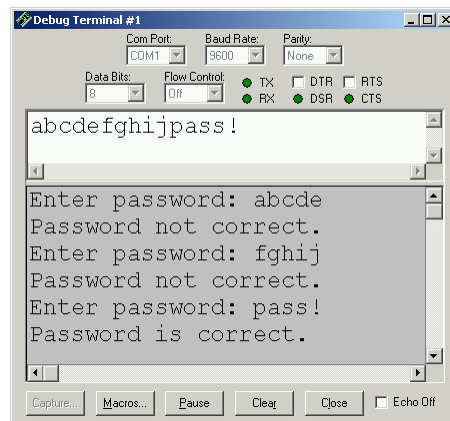



Figura 10-7
Digitazione della Password nella finestra di trasmissione

Clickate nella finestra di trasmissione (superiore) e digitate la password.

10

```
' Che cosa è un Microcontrollore- PasswordChecker.bs2
' Controlla la password digitata nella finestra di trasmissione del
' Terminale di Debug.

' {$STAMP BS2}
' {$PBASIC 2.5}

Password       DATA    "pass!"          ' Memorizza qui la password "segreta".
```

```
index          VAR     Nib       ' Variabile Index.
temp           VAR     Byte      ' Memorizza il carattere singolo.
userEntry      VAR     Byte(5)   ' Memorizza la password inserita dall'utente.

DO

  DEBUG "Enter password: "       ' Istruzioni Utente.

  DEBUGIN STR userEntry \5       ' Prende la password inserita dall'utente.

  FOR index = 0 TO 4                         ' Confronta l'array con DATA
    READ Password + index, temp  ' Prende il prossimo carattere della password
    IF temp <> userEntry(index) THEN EXIT    ' Confronta l'inserimento utente,
  NEXT                                       ' esce se non è uguale.

  IF index <> 5 THEN                       ' Se exit, allora index non è uguale
    DEBUG CR,"Password not correct.", CR   ' a 5 ed il passaggio non è giusto.
  ENDIF

LOOP UNTIL index = 5                         ' Esce dal ciclo solamente quando
                                             ' index = 5.

DEBUG CR, "Password is correct;", CR,        ' Quando la password è giusta
          "program can continue..."          ' Il Programma può continuare.
END
```

Il Vostro Turno – Modificare le Password

- √ Modificate la direttiva **`Password DATA`** in modo che usi una password con cinque differenti caratteri.
- √ Cambiando cinque diversi valori nel programma, potete anche modificarlo in modo che accetti una password di quattro caratteri invece che di cinque caratteri.

Modificare il controllore di Password per l'uso in un programma più grande

L'obiettivo è inserire facilmente il programma PasswordChecker.bs2 nel programma esempio dell'esercizio #3. Vi aiuteranno due cose. Primo, il programma di controllo della password dovrebbe essere modificato in modo che faccia la maggior parte del suo lavoro all'interno di una subroutine. Le differenti parti del programma dovrebbero inoltre essere etichettate con intestazioni commentate in modo di facilitare la commistione dei due programmi.

Programma di Esempio: ReusablePasswordChecker.bs2

ReusablePasswordChecker.bs2 Funziona nello stesso modo di PasswordChecker.bs2, ma per fare il lavoro, usa una subroutine ed è stato riorganizzato in sezioni etichettate.

√ Verificate che il programma funzioni ancora come PasswordChecker.bs2.
√ Esaminate come è stato messo in una subroutine il blocco di codice **DO…LOOP UNTIL**.

```
' -----[ Title ]-----------------------------------------------------------
' Che cosa è un Microcontrollore- ReusablePasswordChecker.bs2
' Controlla la password inserita nella finestra di trasmissione
' del Terminale di Debug.

' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ DATA Directives ]-------------------------------------------------

Password       DATA    "pass!"               ' Store "secret" password here.

' -----[ Variable Declarations ]-------------------------------------------

index          VAR     Nib                   ' Index variable.
temp           VAR     Byte                  ' Stores single char.
userEntry      VAR     Byte(5)               ' Store user entered password.

' -----[ Initialization Routine ]------------------------------------------

GOSUB Check_Password

' -----[ Main Routine ]----------------------------------------------------

' There is no main routine in this program.

DEBUG CR, "All Done"

END

' -----[ Subroutine – Check for Correct Password ]------------------------

Check_Password:

  DO

    DEBUG "Enter password: "                 ' User instructions.

    DEBUGIN STR userEntry \5                 ' Get user input password.
```

```
    FOR index = 0 TO 4                      ' Check array against DATA
      READ Password + index, temp           ' Get next password char
      IF temp <> userEntry(index) THEN EXIT ' Compare to user input,
    NEXT                                    ' exit if not equal.

    IF index <> 5 THEN                      ' If exit, then index not equal
      DEBUG CR,"Password not correct.", CR  ' to 5 and pass is not correct.
    ENDIF

  LOOP UNTIL index = 5                      ' Only get out of loop when
                                            ' index = 5.

  DEBUG CR, "Password is correct."          ' Program can move on when
                                            ' password is correct.

RETURN                                      ' Return when pass is correct.
```

Argomento avanzato: Il Vostro Turno – Commistione dei due Programmi

Ora che ambedue, TerminalOperatedSensorArray.bs2 e ReusablePasswordChecker.bs2 sono stati collaudati, il compito è di combinarli insieme. Il programma finale, dovrebbe controllare la password prima di permettervi di scegliere quale sensore leggere con il Terminale di Debug.

√ Potete aprire nell'Editor del BASIC Stamp tutti e due i programmi (ReusablePasswordChecker.bs2 e TerminalOperatedSensorArray.bs2).

√ Salvate TerminalOperatedSensorArray.bs2 con il nuovo nome di PasswordedSensorTerminal.bs2

Dovrete passare dall'uno all'altro con il tasto TAB per copiare ciascuna sezione di ReusablePasswordChecker.bs2 ed incollarla in PasswordedSensorTerminal.bs2.

√ Copiate la direttiva **`Password DATA`** (inclusa l'intestazione commentata con la linea tratteggiata) da ReusablePasswordChecker.bs2 nel programma PasswordedSensorTerminal.bs2 subito prima della sezione **`I/O Definitions`**.

√ Copiate ed incollate la dichiarazione delle variabili dal programma ReusablePasswordChecker.bs2 nel programma PasswordedSensorTerminal.bs2. Questa dichiarazione delle variabili viene aggiunta a quella già presente in PasswordedSensorTerminal.bs2, senza preoccuparvi di copiare ed incollare l'intestazione commentata **`Variables`**.

√ Copiate la sezione **`Initialization`** (inclusa l'intestazione commentata con la linea tratteggiata) da ReusablePasswordChecker.bs2, ed incollatela tra la sezione **`Variables`** e la **`Main Routine`** in PasswordedSensorTerminal.bs2.

√ Copiate tutta la sezione **`Subroutine`** da ReusablePasswordChecker.bs2 ed incollatela dopo l'ultima subroutine in PasswordedSensorTerminal.bs2.

√ Collaudate PasswordedSensorTerminal.bs2, e vedete se funziona, ricercate e correggete errori se necessario.

SOMMARIO

Questo Capitolo ha spiegato la tecnica di collaudo individuale ciascun sottosistema prima di integrarlo in un sistema più grande. Questo Capitolo ha anche spiegato il comando **ON…GOSUB**, che è particolarmente utile nei sistemi a menu. È stata dimostrata la utile direttiva **PIN** come metodo per assegnare un nome i vostri pin I/O e quindi lasciare all'Editor del BASIC Stamp decidere se leggere un ingresso o scrivere un'uscita. È stato usato un programma password per presentare gli insiemi di variabili, il comando **EXIT,** ed il formattatore **STR** del comando **DEBUGIN**. Il programma password è stato successivamente integrato in un programma più grande per la gestione di un terminale di sensori per dargli maggiore funzionalità.

Domande

1. Quando dovreste collaudare I sottosistemi individualmente prima di provare a farli funzionare insieme? Perché?
2. Quanti programmi degli altri capitoli avete usato in questo capitolo?
3. Quali sono le differenze tra i comandi **GOSUB** ed **ON…GOSUB**?
4. In cosa differisce la direttiva **PIN** da le direttive **CON** e **VAR**?
5. Qual è la differenza fra **EXIT** ed **END**?
6. Come dichiarate un insieme di variabili?
7. Come accedete ad un particolare elemento in un insieme di variabili?

Esercizi

1. Spiegate come usare un ciclo **FOR…NEXT** con un comando **ON…GOSUB** per scorrere attraverso un elenco di subroutine.
2. Spiegate come aggiungere una voce di menu al programma TerminalOperatedSensorArray.bs2 che consenta all'operatore l'opzione di emettere un bip con un altoparlante piezo.
3. Descrivete il processo che usereste (mantenendo la regola del collaudo individuale) per aggiungere un circuito con altoparlante piezo al vostro progetto.
4. Modificate PasswordChecker.bs2 in modo che nel Terminale di Debug appaia il messaggio "avete inserito:" insieme con il testo della password.

Progetti

1. Aggiungete e collaudate un circuito con altoparlante piezo all'insieme di sensori che avete sviluppato in questo capitolo.

2. Modificate TerminalOperatedSensorArray.bs2 in modo che abbia una 5[a] voce di menu che faccia emettere ad un altoparlante piezo un bip a 2 kHz per 1,5 secondi.
3. Modificate TerminalOperatedSensorArray.bs2 in modo che controlli ciascun sensore 10 volte prima di tornare al menu principale. Per le misure RC-time, eliminate tutti i comandi pausa della subroutine. Cambiate le pause della subroutine del tasto con **PAUSE 200**.
4. Se avete eseguito con successo l'Esercizio #4 della sezione Il Vostro Turno: Immaginate che il vostro insieme di sensori sia parte di un sistema di allarme. Immaginate anche che i circuiti tasto siano sensori che rivelino se una porta sia stata aperta o meno. Espandete il menu in PasswordedSensorTerminal.bs2 così che l'utente possa attivare e disattivare l'allarme con una password. Usate dei LED per indicare se l'allarme è attivato o disattivato, ed usate un altoparlante piezo come allarme. Quando l'allarme è attivato, ed un tasto viene premuto e rilasciato, l'altoparlante piezo deve emettere brevi trilli di avviso ogni cinque o dieci secondi per trenta secondi. Se entro trenta secondi la password non viene inserita, l'altoparlante piezo dovrà emettere un suono continuo, ad alto volume ed insistente.
5. Mentre stavate lavorando su qualcuno degli esercizi e progetti di questo libro, potreste aver pensato, "hey, Potrei usarlo per >> inserite qui il vostro progetto: ________________ <<!" Usate le cose che avete appreso in questo libro per inventare un gadget od un aggeggio di vostra invenzione.

10

Ulteriori Approfondimenti

Tutti I libri elencati sono disponibili gratuitamente al sito www.parallax.com. Le versioni citate di seguito sono aggiornate alla data di questa ristampa, ma controllate comunque sul sito www.parallax.com per le ultime revisioni. Ci sforziamo continuamente di migliorare i nostri testi.

Riferimenti

Questo libro è un riferimento essenziale per tutte le guide per studenti Stamps in Class. È ricco di informazioni sulla serie di microcontrollori BASIC Stamp, sulla Board of Education e sulle altre nostre schede, sull'Editor del BASIC Stamp, e sul nostro linguaggio di programmazione PBASIC.

"*Manuale del BASIC Stamp*", Users Manual, Version 2.0c, Parallax Inc., 2000

Guide per Studenti Stamps in Class:

Questo libro (*Che cosa è un Microcontrollore?*) è il testo introduttivo alle altre nostre guide per studenti Stamps in Class. Dopo aver completato lo studio di questo testo, potrete continuare i vostri studi con qualsiasi delle guide per studenti elencate di seguito. Per un introduzione accurata alle pratiche di progettazione dei moderni dispositivi e macchinari, è fortemente raccomandato lo studio degli esercizi e dei progetti della seguente guida per studenti.

"*Applied Sensors*", Student Guide, Version 2.0, Parallax Inc., 2003
"*Basic Analog and Digital*", Student Guide, Version 2.0, Parallax Inc., 2003
"*Industrial Control*", Student Guide, Version 2.0, Parallax Inc., 2002
"*Robotics!*", Student Guide, Version 1.5, Parallax Inc., 2000

Altri Kit di Robotica:

Anche la guida per studenti Stamps in Class *Robotica!* È un introduzione, e dopo averla completata, sarete pronti per uno dei seguenti testi di robotica e kit più avanzati:

"*Advanced Robotics: with the Toddler*", Student Guide, Version 1.2, Parallax Inc., 2003
"*SumoBot*", Student Guide, Version 1.1, Parallax Inc., 2002

Kit di progetti Educazionali:

Elements of Digital Logic ed *Understanding Signals* focalizzano più compiutamente sugli argomenti elettronici, mentre *StampWorks* presenta una varietà di Progetti utili agli hobbisti, inventori e progettisti interessati nel provare una varietà di Progetti.

"*Elements of Digital Logic*", Student Guide, Version 1.0, Parallax Inc., 2003
"*StampWorks*", Manual, Version 1.2, Parallax Inc., 2001
"*Understanding Signals*", Student Guide, Version 1.0, Parallax Inc., 2003

Appendice A: Adattatore USB <-> Seriale

Al momento della stesura di questo scritto, l'adattatore US232B/LC <-> USB costruito dalla Future Technology Devices International è l'adattatore raccomandato per l'uso con i prodotti Parallax. L' US232B/LC viene fornito con l'hardware mostrato nella Figurea A-1 ed un mini-CD ROM con i drivers per l'uso con vari sistemi operativi Microsoft Windows® incluso.

Figurea A-1
L'adattatore US232B/LC USB Seriale della FTDI

Questo adattatore, codice Parallax # 800-00030. viene fornito con CD contenente il software (non mostrato).

Downloads dei Driver Software US232B/LC: I drivers software ed altre informazioni su questo prodotto possono essere scaricati da: http://www.ftdichip.com/FT232.htm.

Appendice B: Equipaggiamenti ed Elenco Componenti

Gli Elenchi Componenti sono suscettibili di variazioni: Prendete nota del fatto che gli Elenchi Componenti e le liste materiali citati in questa Appendice sono suscettibili di variazioni senza avviso. Se avete Domande circa un componente particolare od una quantità, vogliate contattare la Parallax usando il link: www.parallax.com ? Company ? Contact Parallax.

Per completare gli Esercizi di questo libro, dovrete avere una delle seguenti opzioni hardware Parallax:

Opzione 1:

- **Board of Education Full Kit (#28102)** *- E-*
- **Che cosa è un Microcontrollore Parts Kit (#28152 con il libro, #28122 senza il libro)**

Questi due kit sono venduti anche separatamente. La Board of Education Full Kit (il cui contenuto è elencato sotto) è l'equipaggiamento principale del curriculum Stamps in Class, e può essere usato con qualsiasi dei testi e kit Stamps in Class.

Board of Education Full Kit (#28102)		
Parallax Parte #	**Descrizione**	**Quantità**
550-00022	Board of Education	1
800-00016	Ponticelli in filo	10
BS2-IC	modulo BASIC Stamp 2	1
800-00003	Cavo Seriale	1
750-00008	Alimentatore DC– 9 V, 300 mA	1
27000	CD Parallax – Con il Software	1
700-00037	Piedini in gomma – striscia di 4	1

Potete acquistare solamente il kit dei componenti di Che cosa è un Microcontrollore (#28122), oppure i componenti ed il libro stampato *What's a Microcontroller?* (#28152) ovviamente il libro stampato è in Inglese. Questi kit di componenti sono assemblati per

supportare gli esercizi ed i progetti della versione stampata corrente del testo. Il kit dei componenti di Che cosa è un Microcontrollore è elencato nella tabella a pagina seguente.

Kit Che cosa è un Microcontrollore Parts #28122 Che cosa è un Microcontrollore Parts & Text #28152		
Parallax Parte #	**Descrizione**	**Quantità**
150-01020	Resistenza, 5%, 1/4W, 1 kΩ	10
150-01030	Resistenza, 5%, 1/4W, 10 kΩ	4
150-01040	Resistenza, 5%, 1/4W, 100 kΩ	2
150-02020	Resistenza, 5%, 1/4W, 2 kΩ	2
150-02210	Resistenza, 5%, 1/4W, 220 Ω	6
150-04710	Resistenza, 5%, 1/4W, 470 Ω	6
152-01031	Potenziometro - 10 kΩ	1
200-01031	Condensatore, 0.01 μF, 50 V	1
200-01040	Condensatore, 0.1 μF, 100 V	2
201-01080	Condensatore, 1000 μF, 10 V	1
201-03080	Condensatore 3300 μF, 16 V	1
28123	*What's a Microcontroller?* Testo stampato (Incluso solamente in #28152)	1
350-00001	LED - Verde - T1 3/4	2
350-00005	LED - Bi-Colore - T1 3/4	1
350-00006	LED - Rosso - T1 3/4	2
350-00007	LED - Giallo - T1 3/4	2
350-00009	Fotoresistenza	1
350-00027	display LED a 7-segmenti	1
400-00002	Tasto – Normalmente aperto	2
451-00303	Connettore 3 Pin – Maschio/Maschio	1
500-00001	Transistor – 2N3904	1
604-00010	Potenziometro - 10 kΩ Controllato Digitalmente (AD5220-B10)	1
800-00016	Ponticelli da 7,5 cm – Buste di 10	2
900-00001	Altoparlante Piezo	1
900-00005	Servomotore Standard Parallax	1

Opzione 2:

- **Kit BASIC Stamp Che cosa è un Microcontrollore (#90005)**

Questo kit ha le stesse cose di Che cosa è un Microcontrollore Parts & Text, con una HomeWork Board ed accessori che sarebbero altrimenti venduti separatamente. La BASIC Stamp HomeWork Board può essere usata con il testo *Che cosa è un Microcontrollore?* Al posto della Board of Education e del modulo BASIC Stamp 2. La HomeWork Board può essere usata nella maggior parte degli esercizi del curriculum Stamps in Class, sebbene per alcuni esercizi siano necessarie modifiche occasionali ai circuiti. Il kit BASIC Stamp Che cosa è un Microcontrollore include le seguenti cose:

Kit BASIC Stamp Che cosa è un Microcontrollore (#90005)		
Parallax Parte #	**Descrizione**	**Quantità**
555-28158	HomeWork Board con scheda prototipi	1
28123	Testo Che cosa è un Microcontrollore	1
27000	CD Parallax – contiene il software	1
800-00003	Cavo Seriale	1
28122	Che cosa è un Microcontrollore Parts Kit	1
700-00037	Piedini in Gomma – striscia di 4	1

Una Nota per gli Insegnanti: Per l'uso in una classe e per una soluzione economica la HomeWork board è disponibile separatamente in confezioni di 10 pezzi, il costo è significativamente inferiore di una Board of Education + modulo BASIC Stamp 2. Per una quotazione vogliate contattare Parallax Sales Team numero verde (888) 512-1024 (solo per gli Stati Uniti).

BASIC Stamp HomeWork Board Ten-Pack (#28158)		
Parallax Parte #	**Descrizione**	**Quantità**
28158	BASIC Stamp HomeWork Board con scheda prototipi	10

Appendice C: BASIC Stamp e Scheda Madre Componenti e Funzioni

Il BASIC STAMP 2

La Figura C-1 mostra una vista ingrandita del BASIC Stamp 2. I suoi componenti principali e le loro funzioni sono indicate da etichette.

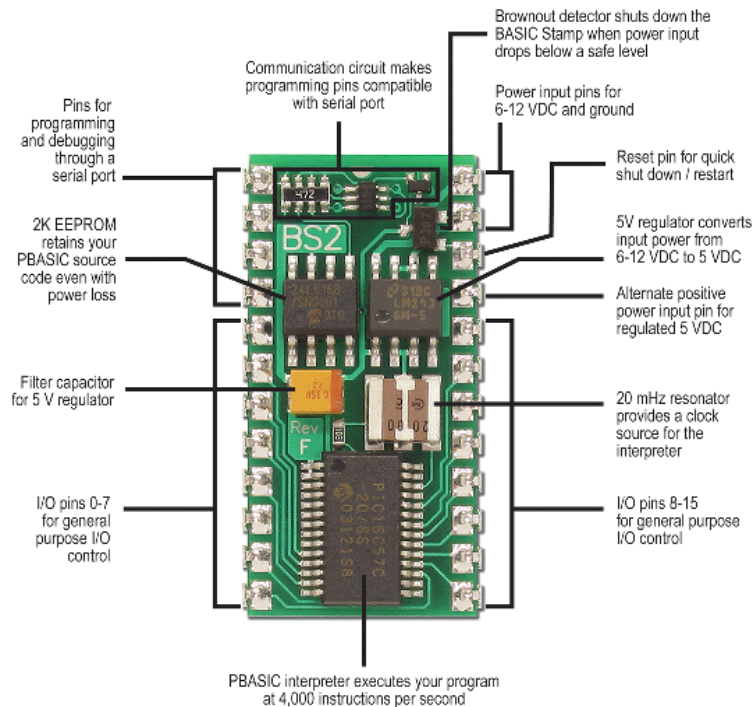



Figura C-1 Componenti e Funzioni del BASIC Stamp 2

La Board of Education Rev C

La Board of Education Rev C è mostrata in Figura C-2 I suoi componenti principali e le loro funzioni sono indicate dalle etichette.

Figura C-2 Board of Education Rev C

La BASIC Stamp HomeWork Board

La BASIC Stamp HomeWork Board è mostrata in Figura C-3. I suoi componenti principali e le loro funzioni sono indicati da etichette.

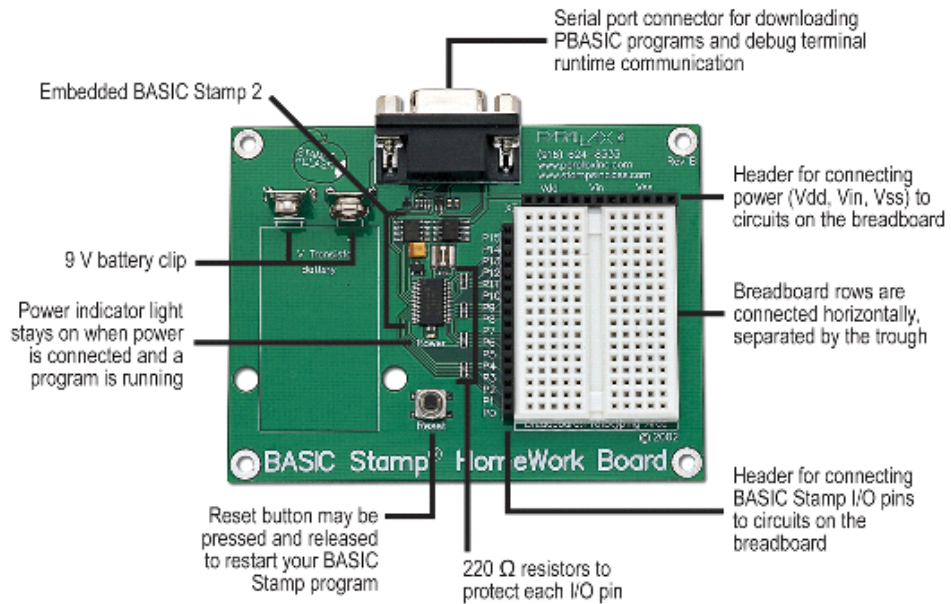



Figura C-3 BASIC Stamp HomeWork Board

La Board of Education Rev B

La Figura C-4 mostra la Board of Education Rev B. I suoi componenti principali e le loro funzioni sono indicate da etichette.

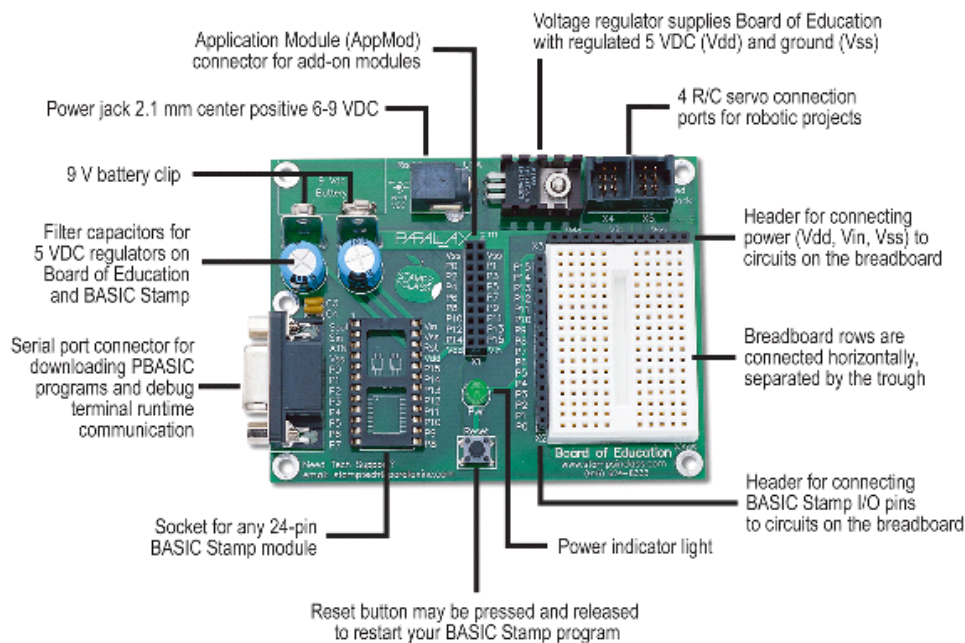



Figura C-4 Board of Education Rev B

ERROR! REFERENCE SOURCE NOT FOUND.

BATTERIE DA 9 V

Per I migliori risultati, Vi raccomandiamo di usare batterie da 9 V.

Specifiche delle batterie da 9 V: Cercate batterie con caratteristiche similari:

Non Ricaricabili

- Alkaline

Ricaricabili

- Ni-Cad (Nickel Cadmio)
- Ni-MH (Nickel Metal Hydride)

Per I migliori risultati, la capacità in milliampere/ora (mAh) dovrebbe essere 100 o più.

Non tutti i carica batterie funzionano per tutti i tipi di batterie. Assicuratevi che il vostro caricabatteria sia adatto per la batteria che state usando (Ni-Cad o Ni-MH).

Seguite attentamente le istruzioni delle batterie e del carica batterie.

ALIMENTATORI DC PARALLAX

La Parallax ha diversi alimentatori che possono essere usati solamente con la Board of Education Rev C. Per gli esperimenti con il Servomotore di questo testo, il ponticello tra i connettori dei Servomotore X4 e X5 deve essere impostato a Vdd. Le alimentazioni elencate nella Tabella D-1 sono progettati per lo standard AC Statunitense, ed ambedue hanno il connettore da 2.1 mm positivo centrale per la connessione al jack della Board of Education.

Tabella D-1: Alimentatori disponibili alla Parallax, Inc.

Parallax Parte #	Ingresso		Uscita	
	VAC	Hz	VDC	mA
750-00008	120	60	9	300
750-00009	120	60	7.5	1000

ALIMENTATORI DC GENERICI

Per i migliori risultati con la BASIC Stamp HomeWork Board o con qualsiasi revisione delle Board of Education, usate un alimentatore DC (chiamato anche alimentatore da muro) con le seguenti caratteristiche:

Ingresso

Dipende dalla nazione in cui vivete e dalla tensione e frequenza disponibile alle vostre prese di corrente. Per l'Europa è 220 VAC, 50Hz, Per gli Stati Uniti ed il Canada, La tensione di ingresso è 120 VAC, 60 Hz.

Uscita

6 VDC, 800 mA
La capacità di corrente può essere maggiore. Per esempio, un alimentatore 6 V, 1000 mA sarebbe ugualmente accettabile.

Connettore Femmina

La Board of Education ha sia il Connettore femmina, che può essere collegato con lo spinotto dell'alimentatore, sia l'attacco per la pila da 9 V, che può essere collegato ad un adattatore di batteria. La HomeWork Board ha solamente l'attacco per la pila da 9 V.

Spinotto

La Figura D-5 mostra un alimentatore comunemente usato con il BASIC Stamp e la Board of Education. Ha uno spinotto da 2.1 mm con positive centrale. Il fatto che il positive sia al centro dello spinotto è adeguatamente rappresentato sull'etichetta.

Figura D-5
Alimentatore DC con spinotto e simbolo di positive centrale.

Adattatore di Batteria 9 V

La Figura D-6 mostra un alimentatore con un adattatore di batteria 9 V che può essere usato con la BASIC Stamp HomeWork Board. Vedere l'avviso della pagina successiva.

Figura D-6
Alimentatore con adattatore di batteria 9 V.

AVVISO – Diffidate degli alimentatori universali con i terminali di alimentazione REVERSIBILI

La Figura D-7 mostra un errore comune che può essere commesso con gli alimentatori universali e che **deve** essere evitato. Molti di questi permettono di invertire i terminali sull'adattatore di batteria 9 V. Sebbene non possa danneggiare il BASIC Stamp, la Board of Education o la Homework Board, può distruggere in pochi secondi i Servomotori Parallax collegati a Vin. Il solo sistema che può proteggere i Servomotori da questo errore è la Board of Education Rev C (con il ponticello messo su Vdd).

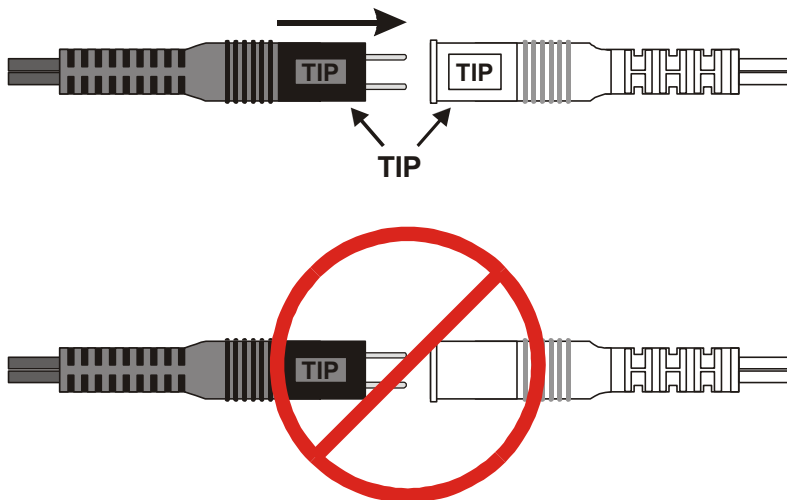



Figura D-7
La Polarità su un alimentatore universale.

Diffidate dei "sostitutori di pile"

Molti "sostitutori di pile" sono progettati per alimentare applicazioni con piccole correnti. Con intensità di corrente dell'ordine dei 10 mA, la loro capacità di corrente è insufficiente per molti degli esercizi di questo testo. Per esempio, due LED collegati a resistenze di 220 Ω assorbono una corrente totale di 14.5 mA, ed il BASIC Stamp ne assorbe dai 3 ai 7 mA. Un Servomotore assorbe fino a 100 mA, e sicuramente non funzionerebbe con un "sostitutore di pile".

NOTA: È molto facile capire se un circuito sta assorbendo più corrente di quanta ne possa fornire l'alimentatore, il LED Pwr sulla Board of Education (o sulla HomeWork Board) lampeggia e/o si attenua.

Appendice E: Ricerca Guasti

Di seguito c'è un elenco di cose che possono essere fatte per risolvere rapidamente qualsiasi difficoltà si incontri nella comunicazione tra l'Editor del BASIC Stamp ed il BASIC Stamp:

Se state usando la Board of Education Rev C, assicuratevi che l'interruttore di alimentazione sia messo nella posizione-1.

Scartate le pile scariche e gli alimentatori non corretti o malfunzionanti, usate una pila da 9 V nuova.

Assicuratevi che il cavo seriale sia ben collegato a tutti e due i connettori, quello della porta COM del computer ed al connettore DB9 sulla Board of Education o sulla BASIC Stamp HomeWork Board.

Assicuratevi che il cavo seriale sia un cavo seriale DIRETTO. NON USATE UN CAVO SERIALE NULL MODEM. La maggior parte dei cavi null modem sono marcati NULL o Null Modem; ispezionate visivamente il cavo per vedere se c'è questa etichetta. Se la trovate, non provate ad usare quel cavo per la programmazione del BASIC Stamp.

Disabilitate qualsiasi software di comunicazione per palmtop.

Se state usando un BASIC Stamp e la Board of Education, controllate anche le cose seguenti:

Assicuratevi che il BASIC Stamp sia inserito correttamente, riferitevi alla Figura 1-30 a pagina 21.

Se state usando un alimentatore DC da rete, assicuratevi che sia correttamente inserito nella presa a 220 V e che lo spinotto sia correttamente inserito nel connettore di alimentazione della Board of Education. Verificate che la luce verde "Pwr" sulla Board of Education emetta luce quando l'alimentatore è in funzione.

Assicuratevi che il BASIC Stamp sia correttamente inserito nel suo zoccolo. Prima di qualsiasi operazione scollegate l'alimentazione, quindi premete a fondo il modulo con le dita. Inoltre ispezionate visivamente il BASIC Stamp per controllare che nessun piedino sia piegato o non inserito nella sua sede del connettore della Board of Education.

Se la finestra di Identificazione è simile a quella mostrata in Figura E-1, significa che l'Editor del BASIC Stamp non riesce a trovare il BASIC Stamp su nessuna delle porte COM. Se avete questo problema, provate i passi seguenti:

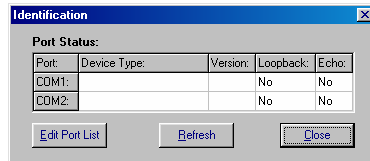



Figura E-1
Finestra di Identificazione

Esempio: BASIC Stamp 2 non trovato sulle porte COM.

Chiudete la finestra di Identificazione.
Assicuratevi che il cavo seriale sia collegato correttamente.
Provate di nuovo il test Run ?   Identify.
Se conoscete il numero di porta COM, ma non appare nella finestra di Identificazione, usate il tasto *Edit Port List* per aggiungere quella porta COM, e quindi provate di nuovo il test Run ?   Identify.
Se avete più di una porta COM, provate a collegare la vostra Board of Education o la BASIC Stamp HomeWork Board ad una porta diversa e controllate se il test Run ?   Identify ora funziona.
Se avete un secondo computer, provate su quest'ultimo.
Se nessuna di queste procedure funziona, andate al sito www.parallax.com e seguite il link Support.

Appendice F: Notizie Ulteriori Circa L’Elettricità

Che cosa è un Elettrone? Un Elettrone è uno delle tre parti fondamentali dell'Atomo; le altre due sono il Protone ed il Neutrone. Uno o più protoni e neutroni stanno insieme al centro dell'atomo e vengono chiamati nucleo. Gli elettroni, in confronto ai protoni ed ai neutroni sono molto piccolo, ed orbitano intorno al nucleo. Gli elettroni si respingono a vicenda, mentre gli elettroni ed i protoni si attraggono.

Che cosa è la Carica? La tendenza di un elettrone ad essere respinto da un altro elettrone ed a essere attratto da un protone vicino, si chiama carica negativa. La tendenza di un protone a respingere un altro protone e ad attrarre un elettrone, è chiamata carica positiva. Quando un atomo ha più elettroni che protoni, viene detto essere a carica negativa. Se un atomo a meno elettroni che protoni, viene detto caricato positivamente. Se un atomo ha lo stesso numero di elettroni e di protoni, viene detto a carica neutra.

Che cosa è la Tensione? La tensione è la differenza di cariche e può essere rappresentata come una "pressione" elettrica. Quando un atomo a carica negativa è vicino ad un atomo a carica positiva, l'elettrone in più dell'atomo a carica negativa cerca di migrare dall'atomo a carica negativa verso l'atomo a carica positiva. Le Batterie sono formate da un composto con cariche negative e da un composto con cariche positive tenute separate. Ciascuno di questi composti è collegato ad uno dei terminali della batteria; il composto caricato positivamente è collegato al terminale positivo (+), ed il composto caricato negativamente è collegato al terminale negativo (-).

Il voltaggio è la misura della pressione elettrica, ed è abbreviato con la lettera 'V' maiuscola. Avete già esperienza con le pile da 9 V usate per alimentare la Board of Education o la HomeWork Board. Altre batterie comuni includono le batterie da 12 V usate nelle auto e le pile da 1,5 V (di tipo AA, AAA, C o D) usate nelle calcolatrici, i video giochi, le torce ed altri dispositivi.

Che cosa è la Corrente? La corrente è la misura del numero degli elettroni al secondo che passano in un circuito. A volte gli atomi reagiscono in una reazione chimica che sviluppa un composto (a carica neutra). Altre volte, gli elettroni abbandonano gli atomi a carica negativa e si riuniscono agli atomi a carica positive attraverso un circuito come quelli che avete costruito e colllaudato. La lettera più comunemente usata per riferirsi alla corrente negli schemi e nei libri, è la lettera 'I' maiuscola.

Che cosa sono gli ampere? Un ampere è l'unità di misura della corrente, e la sigla per indicare la corrente è la lettera 'A' maiuscola. In riferimento ai circuiti che usate con il BASIC Stamp, un ampere è una grande quantità di corrente. È un'unità di misura conveniente per misurare la corrente che una batteria auto fornisce ai fari, alla ventola di raffreddamento del motore ed a altri utilizzatori ad alta potenza.

Che cosa è la Resistenza? La resistenza è l'elemento in un circuito che rallenta il flusso degli elettroni (la corrente) dal polo negativo di una batteria al polo positivo.

L'ohm è l'unità di misura della resistenza. È gia stata spiegata ed è abbreviata con la lettera greca omega (Ω).

Che cosa è un Conduttore? Un conduttore è un elemento in cui la corrente scorre, il filo di rame non ha quasi resistenza, e viene considerato un buon conduttore.

ESERCIZIO: LEGGE DI OHM, TENSIONE E CORRENTE

Questo Esercizio applica alcune definizioni appena discusse.

Componenti per la dimostrazione della legge di Ohm

(1) Resistenza – 220 Ω (rosso-rosso-marrone)
(1) Resistenza – 470 Ω (giallo-viola-marrone)
(1) Resistenza – 1 kΩ (marrone-nero-rosso)
(1) Resistenza – 2 kΩ (rosso-nero-rosso)
(1) LED – qualsiasi colore

Circuito di prova

Il valore della resistenza di R_i nella Figura F-1 può essere cambiato. Una minor resistenza permette lo scorrimento di più corrente attraverso il LED, per cui si illumina di più. Una maggior resistenza diminuendo la corrente che scorre nel circuito del LED ne causerà l'affievolimento.

√ Ogni volta che modificate il circuito scollegate l'alimentazione della vostra Board of Education o della HomeWork Board.
√ Costruite il circuito mostrato cominciando con una resistenza da 220 Ω.
√ Modificate il circuito sostituendo la resistenza da 220 Ω con una resistenza da 470 Ω. Il LED si è affievolito?
√ Ripetete usando la resistenza da 1 kΩ, quindi la resistenza da 2 kΩ, controllando ogni volta la variazione di luminosità.

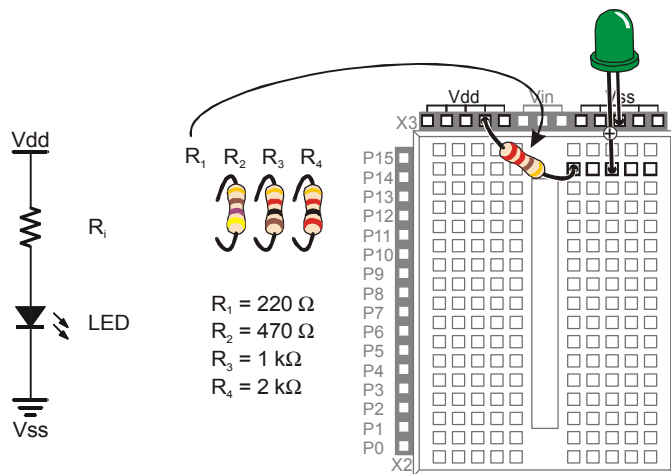



Figura F-1
Dimostrazione della legge di ohm con il LED.

Se state usando una pila da 9-V, potete anche confrontare la luminosità ottenuta con una tensione di alimentazione diversa, Vin. Vin è collegata direttamente al terminale positive della pila da 9 V, e Vss è collegato direttamente al terminale negativo della pila. Vdd è chiamato 5 V stabilizzati. Ed è circa la metà della tensione della pila da 9 V.

- √ Se non state usando una pila da 9 V, scartate questo esercizio ed andate alla sezione per il calcolo della corrente.
- √ Iniziate con il circuito mostrato in Figura F-1, ma usate una resistenza da 1 kΩ.
- √ Fate attenzione alla luminosità del LED.
- √ Togliete l'alimentazione.
- √ Modificate il circuito scollegando la resistenza da Vdd ed inserendola in Vin.
- √ Quando riaccendete, il LED è più luminoso? E di quanto?

NON provate l'esperimento Vin con una resistenza da 220 o da 470 Ω, fornireste al LED una corrente più alta di quella per cui è costruito.

Calcolo della Corrente

Nel Manuale del BASIC Stamp ci sono alcune regole per sapere quanta corrente può essere fornita ai circuiti. Se non seguite queste regole, potete trovarvi con il vostro BASIC Stamp danneggiato. Le regole indicano quanta corrente può fornire un singolo pin I/O e quanta corrente può fornire un gruppo di pin I/O.

Regole della Corrente per i pin I/O del BASIC Stamp

- Un pin I/O può "fornire" 20 mA. In altre parole, se inviate il comando **HIGH** ad un pin I/O, questo non deve fornire al LED più di 20 mA.
- Se modificate il circuito del LED in modo che il BASIC Stamp faccia accendere il LED quando inviate un comando **LOW**, un pin I/O può "ricevere" fino a 25 mA.
- I pin da P0 fino a P7 possono fornire (tra tutti) al massimo 20 mA. Similmente da P8 fino a P15. se avete molti circuiti LED, dovrete usare resistenze di valore più alto in modo di rispettare queste regole.

Se sapete come calcolare quanta corrente userà il vostro circuito, potrete decidere se è corretto per voi far accendere i LED con quella luminosità.

Ogni componente elettronico ha parametri per capire come si comporta con la tensione, la resistenza e la corrente. Per il diodo emettitore di luce, il parametro si chiama tensione diretta del diodo. Per la resistenza, la regola è la legge di Ohm. Questi parametri vi servono per capire quanta corrente userà il vostro circuito LED. Ci sono anche regole per capire come si sommano la tensione e la corrente nei circuiti. Si chiamano teoremi di Kirchoff.

Vdd – Vss = 5 V La tensione (pressione elettrica) tra Vdd e Vss è 5 V. viene chiamata tensione stabilizzata, e funziona come una pila che sia esattamente 5 V.

Vin – Vss = 9 V Se state usando una pila da 9 V, la tensione tra Vin e Vss è 9 V. State attenti. Se state usando un alimentatore da rete, anche se c'è scritto 9 V, può arrivare anche fino a 18 V.

Ground e/o riferimento di massa sono parole che vengono usate riferendosi al terminale negative di un circuito. Parlando del BASIC Stamp e della Board of Education, Vss viene considerato il riferimento di massa. È zero volt, e se state usando una pila da 9 V, corrisponde al terminale negativo di quella pila. Il terminale positivo è 9 V. Vdd è 5 V (al di sopra del riferimento di Vss di 0 V), ed è una tensione speciale create da un circuito integrato che regola (stabilizza) la tensione di alimentazione del BASIC Stamp.

Legge di Ohm: V = I × R La tensione misurata ai capi di una resistenza (V) è uguale alla corrente (I) che scorre nella resistenza moltiplicata per il valore della resistenza(R).

Tensione Diretta del Diodo: Quando un LED emette luce**,** la tensione misurata ai suoi capi sarà circa 1,6 V. indipendentemente dalla corrente che scorre nel LED, che sia grande o piccola, la tensione continuerà ad essere circa 1.6 V.

Teorema della tensione di Kirchoff semplificato: la tensione usata è uguale alla tensione fornita. Se alimentate un circuito con 5 V, le tensioni di tutti I componenti assommeranno sempre a 5 V.

Teorema della corrente di Kirchoff semplificato: la corrente in ingresso è uguale alla corrente in uscita. La corrente che entra in un circuito LED da Vdd ha la stessa intensità di quella che esce da Vss. Inoltre, se collegate tre LED al BASIC Stamp, e ciascun LED assorbe 5 ma, significa che il BASIC Stamp dovrà fornire a tutti i circuiti una corrente totale di 15 mA.

Esempio di Calcolo: Un Circuito, Due Circuiti

Per calcolare quanta corrente assorbe un circuito LED bisogna eseguire due passi:

1. Trovate la tensione ai capi della resistenza.
2. Usate la legge di Ohm per calcolare la corrente che attraversa la resistenza.

La Figura F-2 mostra come trovare la tensione ai capi della resistenza. La tensione di alimentazione è sulla sinistra; ed è 5 V. Le tensioni usate sono sulla destra. All'inizio, non conosciamo la V_R, la tensione ai capi della resistenza. Ma, noi sappiamo che la tensione ai capi del LED è 1,6 V (la tensione diretta del diodo). Sappiamo anche che la tensione ai capi del circuito deve essere 5V a causa della legge della tensione di Kirchoff. La differenza tra 5V e 1,6V è 3,4V, così questa deve essere la tensione ai capi della resistenza V_R.

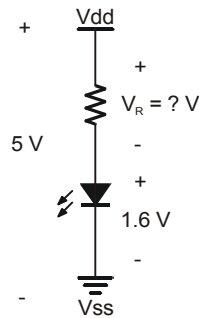



Figura F-2
Le Tensioni ai capi del Circuito, Resistenza, e LED

Kilo è il termine metrico per 1000. Il termine metrico di indicare 1000 è kilo, ed è abbreviato con la lettera k minuscola. Invece di scrivere 1000 Ω, potete scrivere 1 kΩ. e si pronuncia un-kilo-ohm. Similmente, 2000 Ω è scritto 2 kΩ.

Milli è il termine metrico per 1/1000, ed è abbreviato con la m minuscola. Se il BASIC Stamp alimenta un circuito a LED con 3.4 millesimi di ampere, si scrive 3.4 milliampere, o 3.4 mA.

Che cosa è un mA? Pronunciato milliampere, è la notazione metrica di un-millesimo-di-apere. La 'm' in mA è la notazione metrica per milli, che sta per 1/1000. La 'A' in mA sta per ampere. Metteteli insieme ed avrete i milliampere, cosa molto utile per descrivere le quantità di correnti assorbite dal BASIC Stamp e dai suoi circuiti collegati.

Quanta corrente è 7.23 mA? È la quantità di corrente di transito nel LED mostrato a destra nella Figura F-3. Potete sostituire la resistenza da 470 Ω con una da 220 Ω, ed il circuito farà transitare 15.5 mA, ed il LED sarà più luminoso. Se usate una resistenza da 1000 Ω, nel circuito transiteranno 3.4 mA, ed il LED diminuirà la luminosità. Una resistenza da 2000 Ω farà abbassare ulteriormente la luminosità del LED, e la corrente sarà 1.7 mA.

La Figura F-3 mostra un esempio di come calcolare la corrente usata dal circuito con una resistenza da 470 Ω. Iniziate con la legge di Ohm. Saprete le risposte per V (3.4 V) ed R (470 Ω). Ora, tutto quello che dovete fare è risolvere la formula per trovare I (la corrente).

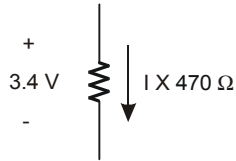



Figura F-3
La Corrente che transita nella Resistenza

Si, è vero - 1 A = 1 V/W (Un ampere è un volt per ohm).

Diciamo che avete due LED accesi contemporaneamente. Questo significa che il BASIC Stamp, sta alimentando I circuiti come mostrato in Figura F-4. Abbiamo ecceduto il limite di corrente di 20 mA? Scopriamolo. Ricordate che la formula semplificata della corrente di Kirchoff dice che la corrente totale assorbita dal circuito è uguale alle correnti che alimentano gli altri circuiti. Questo significa che la I in Figura F-4 deve eguagliare il totale delle due correnti assorbite dai due circuiti. Semplicemente sommate I due assorbimenti, ed avrete la risposta di 14.5 mA. Siete ancora sotto il limite di 20 mA, in questo modo il vostro è un progetto sicuro.

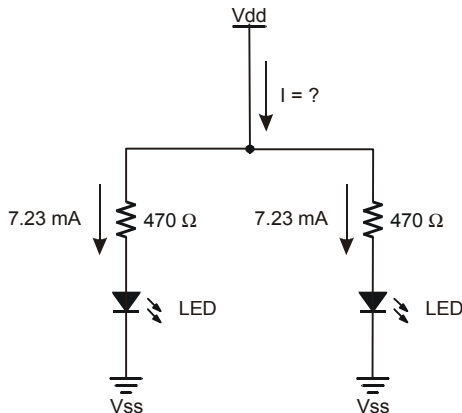



Figura F-4
La Corrente Totale fornita ai due circuiti LED

Il Vostro Turno – Modificare il Circuito

- √ Ripetete l'esercizio in Figura F-2, ma usate Vin – Vss = 9V. La risposta è V_R = 7.4 V
- √ Ripetete l'esercizio in Figura F-3, ma usate una resistenza da 1 kΩ. Risposta: I = 3.4 mA.
- √ Usate V_R = 7.4 V per fare l'esercizio in Figura F-3 con una resistenza da 1 kΩ. Risposta: I = 7.4 mA.
- √ Ripetete l'esercizio in Figura F-4 con una delle resistenze da 470 Ω e l'altra da 1 kΩ. Risposta: I = 7.23 mA + 3.4 mA = 10.63 mA.

Appendice G: Riepilogo del Formato RTTTL

Questo riepilogo intende chiarire alcuni aspetti del formato RTTTL. Le specifiche del "full RTTTL" possono essere trovate in vari siti web. Usando qualsiasi motore di ricerca, usate la parola chiave 'specifiche RTTTL', per trovare le pagine web che contengono le specifiche.

Qui c'è un esempio di suoneria in formato RTTTL:

```
TakeMeOutToTheBallgame:d=4,o=7,b=225:2c6,c,a6,g6,e6,2g.6,2d6,p,
2c6,c,a6,g6,e6,2g.6,g6,p,p,a6,g#6,a6,e6,f6,g6,a6,p,f6,2d6,p,2a6
,a6,a6,b6,c, d,b6,a6,g6
```

Il testo prima dei due punti è ciò che il cellulare visualizza come nome della canzone. In questo caso, la suoneria viene chiamata:

```
TakeMeOutToTheBallGame:
```

Tra il primi ed i secondi due punti, vengono dichiarati i valori di iniziali usando d,o,b e questi significano:

```
d – durata
o – ottava
b – battute per minuto o Tempo.
```

In TakeMeOutToTheBallGame, le impostazioni iniziali sono:

```
d=4,o=7,b=225:
```

Le note nella melodia vengono inserite dopo il secondo due punti, e vengono separate con virgole. Se fossero inserite solo le note, quella nota sarebbe suonata per la durata iniziale, nell'ottava iniziale. Per esempio, la seconda nota in TakeMeOutToTheBallGame è:

```
,c,
```

Dal momento che non ci sono altre informazioni, sarà suonata per la durata di un quarto impostato inizialmente (d=4), nella settima ottava (o=7).

Una nota può avere fino a cinque caratteri fra una virgola e l'altra; Questi è ciò che i caratteri specificano:

```
,durata  nota  diesis  punteggiata ottava,
```

Per Esempio:

`,2g#.6,`

significa suona una G-diesis per una durata di 1 ½ la durata di due quarti, e suonala nella sesta ottava.

Qui ci sono alcuni esempi tratti da TakeMeOutToTheBallGame:

`,2g.6,` – due quarti, G, punteggiata, sesta ottava
`,a6,` – durata di default un quarto, nota A suonata in sesta ottava
`,g#6,` – durata di un quarto, nota g, diesis (scritto #), sesta ottava

Il carattere:

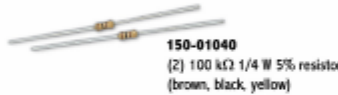
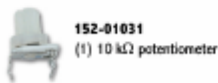
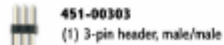
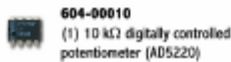
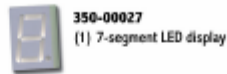
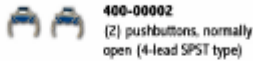
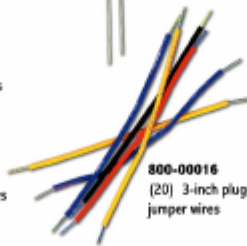
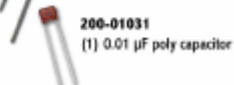
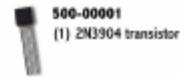
`,p,`

sta per pausa, ed è usato per le pause. Senza altre informazioni, la p dura per la durata di default di un quarto. Potete anche fare una pausa di due quarti usando:

`,2p,`

Qui c'è un esempio di pausa punteggiata di due quarti:

`,2p.,`

In questo caso la pausa durerà per due quarti più un quarto.

150-02020
(2) 2 kΩ 1/4 W 5% resistors
(red, black, red)
150-02210
(6) 220 Ω 1/4 W 5% resistors
(red, red, brown)
150-04710
(6) 470 Ω 1/4 W 5% resistors
(yellow, violet, brown)
150-01040
(2) 100 kΩ 1/4 W 5% resistors
(brown, black, yellow)
150-01030
(4) 10 kΩ 1/4 W 5% resistors
(brown, black, orange)
150-01020
(10) 1 kΩ 1/4 W 5% resistors
(brown, black, red)
152-01031
(1) 10 kΩ potentiometer
451-00303
(1) 3-pin header, male/male
604-00010
(1) 10 kΩ digitally controlled potentiometer (AD5220)
350-00005
(1) bi-color LED
350-00027
(1) 7-segment LED display
400-00002
(2) pushbuttons, normally open (4-lead SPST type)
350-00009
(1) photoresistor
201-03080
(1) 3300 µF electrolytic capacitor
800-00016
(20) 3-inch pluggable jumper wires
900-00005
(1) Parallax Standard Servo
350-00007
(2) yellow LEDs
350-00001
(2) green LEDs
350-00006
(2) red LEDs
900-00001
(1) piezospeaker
200-01040
(2) 0.1 µF ceramic capacitor
201-01080
(1) 1000 µF electrolytic capacitor
200-01031
(1) 0.01 µF poly capacitor
500-00001
(1) 2N3904 transistor